Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Étude de faisabilité d'un didacticiel d'acquisition de structures morphosyntaxiques espagnoles

Kinet, Jacques

Award date:
1984

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Institut d'Informatique
21, Rue Grandgagnage
B-5000 NAMUR

ANNEE ACADEMIQUE 1983-84

Etude de faisabilité d'un didacticiel d'acquisition de structures morphosyntaxiques espagnoles.

****

Promoteur : Monsieur Claude CHERTON

Jacques KINET

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Institut d'Informatique
21, Rue Grandgagnage
B-5000 NAMUR

ANNEE ACADEMIQUE 1983-84

Etude de faisabilité d'un didacticiel d'acquisition de structures morphosyntaxiques espagnoles.

****

Promoteur : Monsieur Claude CHERTON

Jacques KINET

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Remerciements.

Mes remerciements vont

à Monsieur Yvan Régimont, professeur d'espagnol à l'Institut Notre-Dame d'Anderlecht, qui a proposé le sujet de ce mémoire me consacrant beaucoup de son temps afin de m'expliquer ses desiderata et discuter de mes propositions.

à Monsieur Claude Cherton sans qui la rencontre avec Monsieur Régimont eût été impossible et qui m'a aidé de ses conseils et de ses critiques, tout au long de ce travail.

à David Gouthière pour sa collaboration dans la réalisation d'une partie de ce mémoire.

à Mademoiselle Marie-Dominique Delcommune, professeur de français, dont l'aide me fut précieuse dans la conception du chapitre V de ce mémoire concernant la génération de phrases, plus particulièrement dans la détermination des traits de sous-catégorisation et la définition des classes de mots, et qui a assuré la plus grande partie de la dactylographie.

Avertissement.

Ce mémoire se compose de deux volumes : le mémoire proprement dit et les annexes.
Ces dernières traitent plus particulièrement de la réalisation de principes généraux énoncés dans le mémoire. Elles regroupent également des exemples de ces principes.

INTRODUCTION.

Le présent mémoire a pour objet la conception et la réalisation d'un système d'enseignement assisté par ordinateur. Ce système est destiné à l'apprentissage de structures morphosyntaxiques espagnoles incluant les prépositions de lieu "A" et "EN". Il a été demandé par Monsieur Yvan Régimont, professeur d'espagnol dans un établissement d'enseignement secondaire de l'agglomération bruxelloise. C'est à lui que nous sommes redevables de l'aspect didactique et des dialogues avec l'utilisateur du système. Ces deux aspects seront abordés plus loin.

Cette introduction traitera des objectifs du didacticiel, situera le contexte de développement et d'utilisation et enfin présentera le contenu du présent mémoire.

1. Objectifs du didacticiel.

Monsieur Régimont a remarqué que l'acquisition des structures morphosyntaxiques espagnoles incluant les prépositions de lieu "A" et "EN" posait des problèmes à la plupart de ses élèves. La faute la plus fréquente est de confondre les prépositions françaises et espagnoles.
Or, il faut savoir que, si en français la préposition "à" exprime aussi bien le mouvement (je vais à Namur) que la situation (je travaille à Namur), en espagnol la préposition "**a**" n'exprime que le mouvement (voy a Namur mais trabajo en Namur).
De même, en français, la préposition "**en**" exprime également le mouvement (je vais en Espagne) et la situation (je travaille en Espagne) alors qu'en espagnol, elle n'exprime que la situation (trabajo en España mais voy a España).
Confondant les deux langues, les élèves emploient la préposition "**a**" espagnole pour traduire la préposition "à" française exprimant la situation (je travaille à Namur, trabajo ~~a~~ Namur) et la préposition "**en**" espagnole pour traduire la préposition "en" française exprimant le mouvement (je vais en Espagne, voy ~~en~~ España).
Cela constitue deux emplois aussi incorrects l'un que l'autre.
Malgré la rigueur des règles d'emploi espagnoles, les élèves se trompent encore et toujours.
Mais dans leur "Grammaire Espagnole", Marcel Duviols et Jean Villégier n'écrivent-ils pas : "seul l'usage peut apprendre l'emploi correct des prépositions espagnoles" ?

Monsieur Régimont organise donc de nombreuses séances de rattrapage pour que ses élèves acquièrent par la pratique le bon emploi de ces prépositions.
Malheureusement, la conception et la correction d'exercices sont coûteuses en termes horaires et doivent bien souvent être négligées pour cause de non-disponibilité des enseignants.
De plus, l'assiduité des élèves présents à ces séances, quand il y en a, laisse à désirer.

Devant le succès que remportent les micro-ordinateurs accessibles aux élèves de son établissement, Monsieur Régimont a songé à informatiser ces séances de rattrapage et il nous a demandé de prendre en charge ce travail.

Dans les scénarii de séance qu'il nous a proposés et qui seront détaillés plus loin, Monsieur Régimont a d'autre part inclus des exercices de conjugaison. Les élèves auraient ainsi l'occasion de se familiariser avec celle-ci. De plus, le travail proposé se rapprocherait de cette manière de la pratique quotidienne de la langue espagnole.
Dans cette optique, l'idée d'un exercice de créativité où les élèves inventeraient des phrases espagnoles paraît également très séduisante.
Les phrases auraient une structure identique employant les prépositions de lieu. L'utilisateur écrirait une phrase inventée par lui-même et le système tenterait de la corriger.
Les objectifs pédagogiques du didacticiel seraient donc de faire acquérir par les élèves le réflexe d'emploi correct des prépositions de lieu **"a"** et **"en"**, de les familiariser avec les conjugaisons et le vocabulaire espagnols et de stimuler leur créativité dans cette langue.

A un autre niveau se situe un objectif d'éducation et de formation des enseignants à l'informatique. Le travail effectué, tant l'analyse que la programmation, doit constituer un exemple de démarche de conception de didacticiel pour les enseignants s'intéressant à l'informatique.
Ceux-ci pourront s'en inspirer pour concevoir à leur tour leur propre didacticiel.
Cette intention peut amener des contraintes et des choix. Ainsi, le langage de programmation choisi est le Pascal et non l'Assembleur qui aurait permis une exécution des programmes plus rapide mais qui n'est pas accessible à des non informaticiens.
De même, la structuration et la découpe du système devront être tout particulièrement soignées.

De cette manière, l'enseignant pourra facilement le modifier en vue de l'adapter à ses besoins particuliers. Il se sentira également personnellement impliqué dans la construction du système et sera de ce fait beaucoup plus tenté de l'utiliser dans le cadre de son travail d'enseignement.
En particulier, comme tous les programmes d'ordinateur, les didacticiels sont perfectibles et les améliorations à leur apporter n'apparaissent la plupart du temps que pendant leur utilisation.
Et pour les raisons énoncées plus haut, il serait grandement souhaitable que ce soit l'enseignant utilisateur du didacticiel qui puisse apporter les perfectionnements qu'il jugerait lui-même utiles à son utilisation dans le cadre de son enseignement.

Sur un plan plus général, ce mémoire vise à atteindre un autre objectif : montrer aux enseignants l'intérêt pédagogique de didacticiels de qualité.
On pourrait classer les didacticiels en trois catégories : ceux qui fonctionnent sur grosses machines, ceux proposés par des firmes commerciales et ceux réalisés par des enseignants ou en collaboration avec eux.
Cette partition n'est pas exclusive.
Les didacticiels de la première catégorie sont difficilement accessibles aux enseignants du secondaire.
De même, dans l'état actuel des choses, ceux de la deuxième catégorie sont souvent didactiquement faibles.
Enfin, on peut dire que les troisièmes sont assez souvent bien conçus. Néanmoins, ils souffrent d'un mal organique. En effet, on sait difficilement qu'ils existent, à quelles matières ils s'intéressent et où se les procurer. De plus, ils sont peu robustes et leur portabilité est très limitée.
Pour toutes ces raisons, la majorité des enseignants ont un contact plutôt décevant avec l'informatique à l'école.
Nous espérons, par ce travail et par d'autres analogues, montrer aux enseignants l'attrait de didacticiels de qualité dans l'exercice de leur métier.

2. Contexte de développement et d'utilisation.

En vue de situer le contexte de développement et d'utilisation du système, il serait bon de présenter Monsieur Régimont et de préciser son rôle durant ce travail. Il serait également intéressant de définir le profil des élèves appelés à se servir du didacticiel, de déterminer les modalités d'utilisation de celui-ci et de présenter le matériel employé.

Monsieur Régimont nous a donc proposé, à Monsieur Cherton et à moi--même, de réaliser un logiciel d'enseignement d'espagnol.
Plus particulièrement, il nous a présenté cinq catégories d'exercices axés sur l'emploi des prépositions de lieu **"a"** et **"en"**.
Ces exercices seront détaillés plus loin au chapitre 2. Il désirait que nous concevions un système d'exercices assistés par ordinateur.
Il a guidé des ses conseils la conception de différents scénarii et a fait un choix parmi ceux-ci.
Il a également collaboré efficacement à l'élaboration d'un module détaillé au chapitre 6 : le module de génération de phrases.
Mais nous y reviendrons à ce moment.
Son aide me fut vraiment très précieuse et je tiens ici à le remercier de m'avoir consacré tant de temps et d'attention.
Monsieur Régimont enseigne l'espagnol comme troisième langue à des élèves de quatrième année de l'enseignement secondaire.
L'âge moyen de ces élèves est de quinze ans.
Il s'agit de leur première année d'apprentissage de l'espagnol, à raison de quatre heures par semaine. La règle d'emploi des prépositions de lieu leur est enseignée fin octobre, début novembre, soit après une trentaine d'heures de cours. C'est à ce moment qu'ils seront appelés à se servir du didacticiel.
Une remarque intéressante de Monsieur Régimont à propos de ses élèves : ceux-ci passent leur mercredi après-midi et leur samedi dans des galeries commerciales ou dans des boutiques de micro-ordinateurs où ils jouent à des jeux vidéos.
Il ne seraient donc pas dépaysés en utilisant le didacticiel. Ils pourraient même le considérer comme un jeu, un de plus.

En ce qui concerne l'utilisation par les élèves du didacticiel, elle doit se faire de manière autonome, sans la présence du professeur.
Néanmoins, un certain contrôle du travail accompli par les élèves doit pouvoir être exercé par le professeur.
Pour cela, le système donne la possibilité d'une évaluation globale représentative des exercices effectués.
Faute de mieux, cette évaluation se fera sous forme d'un "score" établi en fonction des réponses de l'élève aux questions posées par le didacticiel. Le professeur pourra prendre connaissance des "scores" des différents élèves ayant accompli une session après celle-ci. De même, l'élève utilisateur pourra consulter le résultat qu'il a atteint à tout moment de la session. A la fin de celle-ci, il pourra également savoir ce que représente ce résultat en valeur relative par rapport à d'autres ou en comparaison avec les exigences du professeur.

L'établissement scolaire dans lequel Monsieur Régimont enseigne dispose de micro-ordinateurs Apple II 48 K. basic avec carte 80 colonnes. C'est également sur Apple II qu'ont été développés les modules réalisés dans le cadre de ce mémoire.
Les programmes ont été écrits en Pascal.
Ne disposant pas des lettres minuscules sur les Apple II disponibles à l'Institut, les algorithmes n'ont pas tenu explicitement compte de la différence entre minuscules et majuscules ainsi que des différents accents pouvant être utilisés en espagnol.
Toutefois, les modifications à apporter aux programmes pour traiter majuscules, minuscules et accents sont minimes et très localisées et la structure des modules ne devra pas être revue.

Le développement des programmes sur Apple II m'a posé des problèmes particuliers, inhérent à la machine et au système d'exploitation U.C.S.D.
On peut citer par exemple la taille des fichiers limitée à 36 blocs qui m'a forcé à disperser un seul gros programme sur plusieurs fichiers et même plusieurs diskettes.
Il est indéniable que mon manque total d'expérience du type de matériel et mon habitude limitée d'une telle taille d'application ont sérieusement retardé l'avancement de mon travail.
Avec l'expérience accumulée tout au long de l'élaboration de ce mémoire et si je devais le refaire, j'irais beaucoup plus vite.

3. Contenu du mémoire.

Dans le cadre de ce mémoire, vu le temps qui m'était imparti, il m'était impossible de réaliser l'ensemble du didacticiel.
Aussi, j'ai dû me limiter à quatre objectifs précis et distincts :
- l'élaboration d'un scénario pour le didacticiel tout entier, avec l'aide précieuse de Monsieur Régimont, scénario s'appuyant sur l'enchaînement de grilles d'écran.
- la conception et la réalisation, en collaboration avec David Gouthière, d'un module de correction automatique d'orthographe destiné à être utilisé par le didacticiel.
- la conception et la réalisation d'un module de génération automatique de phrases spécifiques aux exercices retenus pour le didacticiel.
- la conception et la réalisation d'un système de gestion des fichiers passés comme paramètres à ces modules.
Ce système permettra à l'enseignant de déterminer lui-même le vocabulaire espagnol employé par le didacticiel ainsi que des règles liées à la sémantique et permettant au système de générer des phrases acceptables.
A titre d'exemple et afin de vérifier le bien-fondé de mon module de génération de phrases, j'ai moi-même mené à bien cette démarche avec la collaboration active de Monsieur Régimont.
Le didacticiel pourra donc fonctionner avec le vocabulaire et les phrases que Monsieur Régimont et moi-même avons déterminés.
Néanmoins, il est important que l'enseignant puisse adapter le didacticiel à son propre usage et il s'en servira d'autant plus volontiers qu'il a pu le modifier pour qu'il corresponde à ses besoins.

La suite de cet ouvrage est découpée de la façon suivante.

Tout d'abord dans le chapitre 1, seront spécifiées les contraintes posées par quelques règles de grammaire espagnole.

Dans le chapitre 2, le problème sera identifié en exposant les types d'exercices intégrés au didacticiel.

Le chapitre 3 jettera les bases du didacticiel à construire, sous forme de scénario.

Le chapitre 4 expliquera les implications et les raisons de la réalisation des modules de correction d'orthographe et de génération de phrases.

Le chapitre 5 présentera le développement complet, de l'analyse à la mise en oeuvre, du module de correction d'orthographe alors que les chapitres 6 et 7 feront de même respectivement pour le module de génération de phrases et le module de gestion des fichiers.

CHAPITRE I :

SPECIFICATION DES CONTRAINTES : QUELQUES REGLES DE GRAMMAIRE ESPAGNOLE.

Avant de s'attaquer à l'identification du projet, c'est-à-dire ce que l'on voudrait que le didacticiel fasse, il importe de préciser quelques règles de grammaire espagnole auxquelles devra se soumettre le système.
Ces règles sont tirées de deux grammaires :
celle de Marcel DUVIOLS et Jean VILLEGIER (1) et celle de Jean BOUZET (2).

1. Règles d'écriture de phrases.

L'ordre des mots est beaucoup moins important en espagnol qu'en français.
Néanmoins, dans la structure de phrase "Sujet - Verbe - Complément de lieu", il faut que la préposition soit juste devant le complément de lieu.
Les phrases interrogatives ou exclamatives commencent par un point d'interrogation ou d'exclamation renversé.
Le "vous" de politesse se rend par "**usted**" avec la troisième personne du singulier, si l'on s'adresse à une seule personne, par "**ustedes**" et la troisième personne du pluriel si l'on s'adresse à plusieurs personnes.

2. Expression du pronom personnel sujet.

Les terminaisons verbales espagnoles sont assez différentes pour que le pronom sujet ne soit pas nécessaire. Il n'est généralement pas exprimé.
On exprime le pronom sujet seulement dans les cas suivants :

a) Lorsque la clarté l'exige par suite de l'identité de terminaison à la première et à la troisième personne du singulier de certains temps (Imparfait de l'indicatif, conditionnel, temps du subjonctif).

b) Lorsqu'on veut insister sur le sujet ou marquer une opposition.

3. Emploi des articles.

Les formes de l'article défini sont les suivantes :

Masculin :	**el**	le	**los**	les
Féminin :	**la**	la	**las**	les
Neutre :	**lo**	le		

En général, on emploie l'article masculin devant les noms masculins et l'article féminin devant les noms féminins.

On emploie généralement l'article masculin devant les autres mots pris substantivement : adjectifs, adverbes, verbes, participes et mots invariables.
Les formes de l'article indéfini sont les suivantes :
Singulier : **un, una**
Pluriel : **unos, unas**
L'article indéfini s'emploie beaucoup moins en espagnol qu'en français ... Les formes indéfinies françaises "du, de la, des" ne se traduisent pas.
On emploie l'article devant les mots "**señor**", monsieur, "**señora**", madame, "**señorita**", mademoiselle, suivis d'un nom propre ou d'un titre. En revanche, il n'y a pas d'article devant le titre.
On emploie aussi l'article devant les noms de jours de la semaine quand les mots "**pasado**" ou "**último**", dernier, "**proximo**" ou "**que viene**", prochain, sont exprimés ou sous-entendus.
Devant les noms de pays ou de provinces, on n'emploie pas l'article s'ils ne sont pas déterminés par un adjectif ou un complément.
On construit sans article les compléments de lieu constitués par les mots "**casa**" (dans toutes les équivalences du français chez), "**misa**", messe et "**clase**", classe, cours, même suivis d'un complément d'ordre général.
Les mots "**paseo**", promenade, "**palacio**" dans le sens de palais royal, "**presidio**", bagne, se construisent également sans article dans les compléments de lieu, mais à condition qu'ils ne soient accompagnés d'aucune autre détermination. "**Caza**" et "**pesca**" n'admettent cette construction qu'avec le verbe "**ir**", aller.
Il n'y a pas de nom neutre en espagnol, mais un article neutre "**lo**" qui s'emploie devant un adjectif, un participe ou un adverbe pris substantivement dans un sens général et abstrait.
Devant un nom féminin singulier commençant par "**a**" (ou "**ha**") tonique, on emploie l'article (masculin) "**el**" (voir en français "mon épée"), et de préférence "**un**" si l'article est indéfini.
Enfin, l'article défini "**el**" se contracte avec la préposition "**a**" pour former "**al**".

4. Les prépositions de lieu.

Il existe une différence fondamentale quant à l'emploi des prépositions de lieu entre le français et l'espagnol : en français, le choix de la préposition dépend souvent du complément qui la suit ; en espagnol, il dépend le plus souvent du verbe qui la précède.
Dans les compléments de lieu, l'espagnol établit une distinction très nette dans l'emploi des prépositions "a" et "**en**", parallèle à celle qui est faite en latin par les questions "**quo** ?" et "**ubi** ?"

Quelle que soit la préposition employée en français (à, en, dans), il faut la rendre par **"a"** après un verbe de mouvement, devant le nom qui en indique le but (question **"quo"**);
par **"en"** si le verbe signifie résidence ou s'il indique une action qui s'exerce dans l'endroit indiqué (question **"ubi"**).

5. Le verbe.

Au point de vue de la conjugaison, les verbes espagnols se répartissent en trois groupes caractérisés par la terminaison de leurs infinitifs : infinitifs terminés en "ar", infinitifs terminés en **"er"** et infinitifs terminés en **"ir"**.
La conjugaison espagnole représente dans ses grandes lignes la conjugaison active du latin et comprend à peu près les mêmes temps et les mêmes modes que le français.
Les terminaisons des verbes varient selon le mode, le temps, la personne et le groupe de conjugaison auquel ils appartiennent.
Les éléments les plus stables sont ceux qui servent à identifier la personne du verbe et qu'on appelle désinences personnelles : ils sont placés à la fin de la terminaison.
Ces éléments se retrouvent à tous les temps et à tous les modes personnels sauf à l'impératif et au passé simple.
Les désinences personnelles sont **"s"** à la deuxième personne du singulier, **"mos"** à la première personne du pluriel, **"is"** à la deuxième personne du pluriel et **"n"** à la troisième personne du pluriel.
Il n'y en a pas à la première et à la troisième personne du singulier, ce qui peut provoquer des confusions entre ces deux formes (comme il a déjà été dit plus haut).
Pour conjuguer un verbe régulier, il suffit, en principe, d'ajouter à son radical les terminaisons propres aux verbes de son groupe.
Les temps composés se forment comme en français, en faisant précéder d'un auxiliaire le participe passé du verbe à conjuguer.
Cet auxiliaire est en espagnol **"haber"**, avoir, pour tous les cas, c'est-à-dire aussi bien pour les verbes transitifs que pour les verbes intransitifs et les verbes pronominaux.
Le participe passé conjugué avec **"haber"** est toujours invariable.
La conjugaison passive se forme avec l'auxiliaire **"ser"** et le participe passé du verbe à conjuguer. Le participe passé s'accorde dans ce cas en genre et en nombre avec le sujet.

Dans la conjugaison pronominale, les formes simples ou composées du verbe sont précédées des pronoms réfléchis **"me"**, **"te"**, **"se"**, **"nos"**, **"os"** ou **"se"**.
Mais à l'infinitif, à l'indicatif, et au gérondif, il se place après et ne forme avec lui qu'un seul mot : c'est l'enclise.
Aux temps composés, le pronom complément se place avant l'auxiliaire.
Mais à l'indicatif passé comme au gérondif composé, il se place après l'auxiliaire.
En principe, dans l'interrogation, le sujet se place après le verbe : **"¿ accepta usted ?"**, acceptez-vous ?; **"¿ vendrá tu amigo ?"**, Ton ami viendra-t-il ?
Mais comme d'une part, les pronoms sujets sont habituellement omis en espagnol, et que d'autre part, l'inversion verbe-sujet peut répondre à d'autres intentions, on peut dire que la conjugaison interrogative n'a pas de forme propre en espagnol et que l'interrogation est surtout marquée par l'intonation pour l'auteur et par les deux signes d'interrogation pour le lecteur.
Quant à la conjugaison négative, elle n'offre aucune particularité si ce n'est la présence de **"no"** devant le verbe, d'ailleurs susceptible d'être remplacé par un autre mot à valeur négative : **"nunca"**, **"nadie"**, **"ninguno"**, **"nada"...**
"No" doit être placé après le sujet et avant les pronoms proclitiques qui précèdent le verbe : **"Tú no me los diras"**, tu ne me le diras pas.
Rappelons que la présence de la négation entraîne la substitution de l'impératif par le subjonctif.

Les tableaux de conjugaison repris à la "Grammaire espagnole" de Marcel Duviols et Jean Villégier sont les suivants.

1. CANTAR.

Indicatif présent	Indicatif imparfait
Canto	Cantaba
Cantas	Cantabas
Canta	Cantaba
Cantamos	Cantábamos
Cantáis	Cantabais
Cantan	Cantaban
Indicatif futur	**Indicatif passé simple**
Cantaré	Canté
Cantarás	Cantaste
Cantará	Cantó
Cantaremos	Cantamos
Cantaréis	Cantasteis
Cantarán	Cantaron

Subjonctif présent	Conditionnel
Cante	Cantaría
Cantes	Cantarías
Cante	Cantaría
Cantemos	Cantaríamos
Cantéis	Cantaríais
Canten	Cantarían
Impératif présent	**Gérondif**
Canta	Cantando
Cante	
Cantemos	**Participe-passé**
Cantad	Cantado
Canten	

2. BEBER.

Indicatif présent	Indicatif imparfait
Bebo	Bebía
Bebes	Bebías
Bebe	Bebía
Bebemos	Bebíamos
Bebéis	Bebíais
Beben	Bebían
Indicatif futur	**Indicatif passé-simple**
Beberé	Bebí
Beberás	Bebiste
Beberá	Bebió
Beberemos	Bebimos
Beberéis	Bebisteis
Beberán	Bebieron
Subjonctif présent	**Conditionnel**
Beba	Bebería
Bebas	Beberías
Beba	Bebería
Bebamos	Beberíamos
Bebáis	Beberíais
Beban	Beberían
Impératif présent	**Gérondif**
Bebe	Bebiendo
Beba	
Bebamos	**Participe-passé**
Bebed	Bebido
Beban	

3. VIVIR.

Indicatif présent	Indicatif imparfait
Vivo	Vivía
Vives	Vivías
Vive	Vivía
Vivimos	Vivíamos
Vivís	Vivíais
Viven	Vivían
Indicatif futur	**Indicatif passé-simple**
Viviré	Viví
Vivirás	Viviste
Vivirá	Vivió
Viviremos	Vivimos
Viviréis	Vivisteis
Vivirán	Vivieron
Subjonctif présent	**Conditionnel**
Viva	Viviría
Vivas	Vivirías
Viva	Viviría
Vivamos	Viviríamos
Viváis	Viviríais
Vivan	Vivirían
Impératif présent	**Gérondif**
Vive	Viviendo
Viva	
Vivamos	**Participe-passé**
Vivid	Vivido
Vivan	

Voici également, d'après les mêmes sources, les conjugaisons des auxiliaires : ser, estar et haber.

1. SER.

Indicatif présent	Indicatif imparfait	Indicatif futur
Soy	Era	Seré
Eres	Eras	Serás
Es	Era	Será
Somos	Éramos	Seremos
Sois	Erais	Seréis
Son	Eran	Serán

Indicatif passé-simple	Subjonctif présent	Conditionnel
Fui	Sea	Ser ía
Fuiste	Seas	Serías
Fue	Sea	Sería
Fuimos	Seamos	Ser íamos
Fuisteis	Seáis	Seríais
Fueron	Sean	Ser ían
Impératif présent	**Gérondif**	**Participe-passé**
Sé	Siendo	Sido
Sea		
Seamos		
Sed		
Sean		

2. ESTAR.

Indicatif présent	Indicatif imparfait	Indicatif futur
Estoy	Estaba	Estaré
Estás	Estabas	Estarás
Está	Estaba	Estará
Estamos	Estábamos	Estaremos
Estáis	Estabais	Estaréis
Están	Estaban	Estarán
Indicatif passé-simple	**Subjonctif présent**	**Conditionnel**
Estuve	Esté	Estar ía
Estuviste	Estés	Estar ías
Estuvo	Esté	Estar ía
Estuvimos	Estemos	Estar íamos
Estuvisteis	Estéis	Estar íais
Estuvieron	Estén	Estar ían
Impératif présent	**Gérondif**	**Participe-passé**
Está	Estando	Estado
Esté		
Estemos		
Estad		
Estén		

3. HABER.

Indicatif présent	Indicatif imparfait	Indicatif futur
He	Había	Habré
Has	Habías	Habrás
Ha	Había	Habrá
Hemos	Habíamos	Habremos
Habéis	Habíais	Habréis
Han	Habían	Habrán

Indicatif passé-simple	Subjonctif présent	Conditionnel
Hube	**Haya**	Habría
Hubiste	**Hayas**	Habrías
Hubo	**Haya**	Habría
Hubimos	**Hayamos**	Habríamos
Hubisteis	**Hayáis**	Habríais
Hubieron	**Hayan**	Habrían

Impératif	Gérondif	Participe-passé
inusité	Habiendo	Habido

Les verbes irréguliers sont assez nombreux, aussi je n'en parlerai pas ici.

Pour les besoins de mon travail, Monsieur Régimont m'a fourni une liste de seize verbes demandant des prépositions de lieu.
Il désirait que les exercices intégrés au didacticiel ne portent que sur ces derniers car ce sont les seuls verbes employant des prépositions de lieu que ses élèves doivent apprendre durant leur première année d'apprentissage de l'espagnol.

En voici la liste :

Comer : manger
Estar : être
Ir : aller, se diriger vers, marcher
Llegar : arriver, parvenir, atteindre
Marchar(se) : partir (verbe pronominal)
Meter(se) : s'introduire (verbe pronominal)
Morir : mourir
Nadar : nager, flotter
Pasar : passer, dans le sens passer un mois (**un mes**), une semaine (**una semana**), les vacances (**las vacaciones**), un moment (**un rato**)
Quedar(se) : rester, demeurer (verbe pronominal)
Quedar : rester
Situar(se) : se situer, se placer (verbe pronominal)
Subir : monter, s'élever
Trabajar : travailler
Venir : venir, arriver
Vivir : vivre.

Parmi ces verbes, "**trabajar**", "**pasar**", "**quedar**" et "**nadar**" se conjuguent régulièrement selon le modèle de la première conjugaison "**cantar**".
"**Meter**" et "**comer**" se conjuguent régulièrement selon le modèle de la deuxième conjugaison "**beber**".
"**Vivir**" et "**Subir**" se conjuguent selon le modèle de la troisième conjugaison qui est justement "**vivir**".
La conjugaison de "**estar**" a déjà été détaillée un peu plus haut à la page
"**Marchar(se)**" et "**quedar(se)**" se conjuguent en gros selon le modèle de la première conjugaison "**cantar**" ; néanmoins, ils sont irréguliers à l'impératif.
De plus, ce sont des verbes pronominaux.
Voici donc leur conjugaison entière avec le pronom complément, sur le modèle de "**situar(se)**".

SITUARSE.

Indicatif présent	Indicatif imparfait	Indicatif futur
me situó	me situaba	me situaré
te situas	te situabas	te situarás
se situa	se situaba	se situará
nos situamos	nos situábamos	nos situaremos
os situáis	os situabais	os situaréis
se situan	se situaban	se situarán

Subjonctif présent	Impératif	Indicatif passé
me situe	sitúate	me situé
te situes	sitúese	te situaste
se situe	situémonos	se situó
nos situemos	situaos	nos situamos
os situéis	situense	os situasteis
se situen		se situaron

"Llegar" se conjugue régulièrement comme **"cantar"** à l'imparfait et au futur de l'indicatif. Pour le reste, il est complètement irrégulier.
En voici la conjugaison : **Llegar**

Indicatif présent	Indicatif imparfait	Indicatif futur
Llego	Llegaba	Llegaré
Llegas	Llegabas	Llegarás
Llega	Llegaba	Llegará
Llegamos	Llegábamos	Llegaremos
Llegaís	Llegabais	Llegaréis
Llegan	Llegaban	Llegarán

Indicatif passé	Subjonctif présent	Impératif
Llegué	Llegue	Llega
Llegaste	Llegues	Llegue
Llegó	Llegue	
Llegamos	Lleguemos	Lleguemos
Llegasteis	Lleguéis	Llegad
Llegaron	Lleguen	Lleguen

"Morir", "Ir" et "Venir" sont complètement irréguliers.
Voici leurs conjugaisons respectives.

MORIR

Indicatif présent	Indicatif imparfait	Indicatif futur
Muero	Moría	Moriré
Mueres	Morías	Morirás
Muere	Moría	Morirá
Morimos	Moríamos	Moriremos
Morís	Moríais	Moriréis
Mueren	Morían	Morirán
Indicatif passé	Subjonctif présent	Impératif
Morí	**Muera**	**Muere**
Moriste	**Mueras**	**Muera**
Murió	**Muera**	
Morimos	**Muramos**	**Muramos**
Moristeis	**Muraís**	**Morid**
Murieron	**Mueran**	**Mueran**

IR

Indicatif présent	Indicatif imparfait	Indicatif futur
Voy	**Iba**	Iré
Vas	**Ibas**	Irás
Va	**Iba**	Irá
Vamos	**Íbamos**	Iremos
Vaís	**Ibais**	Iréis
Van	**Iban**	Irán
Indicatif passé	Subjonctif présent	Impératif
Fui	**Vaya**	**Ve** ou **Vete**
Fuiste	**Vayas**	**Vaya** **Váyase**
Fue	**Vaya**	
Fuimos	**Vayamos**	**Vayamos** **Vámonos**
Fuisteis	**Vayáis**	**Id** **Idos**
Fueron	**Vayan**	**Vayan** **Váyanse**

VENIR.

Indicatif présent	Indicatif imparfait	Indicatif futur
Vengo	Venía	Vendré
Vienes	Venías	Vendrás
Viene	Venía	Vendrá
Venimos	Veníamos	Vendremos
Venís	Veníais	Vendréis
Vienen	Venían	Vendrán
Indicatif passé	**Subjonctif présent**	**Impératif**
Vine	**Venga**	**Ven**
Viniste	**Vengas**	**Venga**
Vino	**Venga**	
Vinimos	**Vengamos**	**Vengamos**
Vinisteis	**Vengáis**	**Venid**
Vinieron	**Vengan**	**Vengan**

6. Formation du pluriel des noms

Pour former le pluriel, on ajoute "s" aux mots terminés par une voyelle atone et on ajoute "es" aux mots terminés par une consonne, l'"y" étant considéré comme une consonne, et aux mots terminés par un "i" accentué.

Les mots terminés par toute autre voyelle accentuée prennent seulement "s" au pluriel, les noms en "a", "o" et "u", généralement d'origine étrangère admettent les deux terminaisons.

Les polysyllables en "s" non accentués sur la dernière restent invariables, de même que les noms propres terminés par "z" ou "s".

Le "z" final devient "c" devant "es".

Sauf quelques exceptions, dans les mots composés de deux termes variables, le second seulement prend la marque du pluriel. Les noms composés où il entre un verbe sont ordinairement invariables.

BIBLIOGRAPHIE DU CHAPITRE 1.

(1) Marcel DUVIOLS, Jean VILLEGIER, Grammaire espagnole, Paris, Hatier, 1964.

(2) Jean BOUZET, Grammaire espagnole, Paris, Belin, 1978

C H A P I T R E I I :

IDENTIFICATION DU PROJET.

Comme il a été dit au deuxième paragraphe de l'introduction, le projet est de concevoir et de réaliser un système d'exercices assistés par ordinateur.
Les exercices que les élèves pourraient exécuter avec l'aide du didacticiel sont à ranger en cinq catégories. Ils ont tous été proposés par Monsieur Régimont.
Après concertation entre Monsieur Régimont, Monsieur Cherton et moi-même, ces cinq types d'exercices ont été retenus pour former l'ossature du logiciel, mais avec quelques variantes pour trois d'entre eux.

1. TYPE 1 : Mettre la bonne préposition.

L'intitulé de l'exercice serait :
"Complétez les phrases suivantes avec les prépositions A/EN".
Sujet Verbe ... Lieu.
Le verbe est donné à la forme correcte.
Plusieurs phrases sont soumises à l'élève et celui-ci doit simplement écrire la préposition qui convient.
Exemple : "Los ninos van ... la calle".
qui devient :
"Los ninos van a la calle".

2. TYPE 2 : Conjuguer correctement le verbe et mettre la préposition qui convient.

L'intitulé de l'exercice serait :
"Conjuguez le verbe à la forme correcte et complétez avec la préposition qui convient".
Sujet infinitif ... Lieu (mode et temps)
préposition
Dans cet exercice, l'élève écrit non seulement la préposition qui convient mais il doit en plus conjuguer le verbe au mode et au temps qui lui sont demandés.
Le verbe est proposé à l'infinitif.
Exemple : "Mi padre estar ... Espana"
(indicatif imparfait)
qui doit devenir
"Mi padre **estaba** en Espana".

3. TYPE 3 : Compléter les phrases par un verbe et une préposition adéquats.

Sujet ... ... Lieu.
Verbe préposition

Dans cet exercice, les phrases proposées à l'élève sont dépourvues de verbe et de préposition.
Il s'agit pour lui de choisir un verbe dans une liste qui lui est soumise, ainsi que la préposition correcte. Il peut arriver que plusieurs verbes de la liste conviennent pour une même phrase. Ces verbes ne se construisent pas nécessairement tous avec la même préposition. Donc, pour une même phrase, plusieurs combinaisons "Verbe de la liste-préposition" peuvent convenir.
Il existe deux variantes pour ce type d'exercice.

3.a. Liste de verbes conjugués.

L'intitulé de l'exercice serait :
"Complétez les phrases suivantes avec un verbe de la liste et une préposition adéquate".
Dans cette variante, les verbes de la liste proposée à l'élève sont conjugués.
Ce dernier n'aurait qu'à choisir un verbe qui convient dans la liste proposée et compléter la phrase avec la préposition correcte.
Exemple : "Los ninos ... ... el patio"
qui doit devenir :
"Los ninos estaban en el patio"
"estaban" se trouvant dans la liste des verbes conjugués proposés.

3.b. Liste des verbes à l'infinitif.

L'intitulé de l'exercice serait :
"Complétez les phrases suivantes avec un verbe de la liste à la forme correspondante et complétez avec une préposition".
Dans cette variante, les verbes de la liste proposée à l'élève sont à l'infinitif.
A la différence de la première variante, l'élève doit ici en plus conjuguer le verbe pour l'intégrer dans la phrase.
Le temps et le mode demandés sont indiqués à côté de la phrase.
Exemple : "Mi padre ... ... Espana".
(indicatif imparfait)
qui doit devenir
"Mi padre estaba en Espana".
avec "estar" se trouvant dans la liste des verbes proposés à l'infinitif.

4. TYPE 4 : Remettre les mots dans l'ordre.

Verbe Lieu Sujet
Lieu Verbe Sujet
Sujet Lieu Verbe

On propose une suite de mots à l'élève. Ce dernier doit réécrire ceux-ci pour former une phrase correcte et mettre la préposition qui convient devant le complément de lieu.

4.a. Verbes conjugués.

L'intitulé de l'exercice serait :
"Formez des phrases avec les mots suivants et complétez avec une préposition".
L'élève doit réécrire les mots dans un ordre correct en y ajoutant la préposition qui convient. Le verbe est déjà conjugué.
Exemple : "el patio - va - mi hermano"
qui doit devenir
" mi hermano va al patio"

4.b. Verbes à l'infinitif sans temps demandé.

L'intitulé de l'exercice serait :
"Formez des phrases avec les mots suivants. Conjuguez correctement le verbe".
Il s'agit ici de la proposition initiale de Monsieur Régimont.
L'élève doit réécrire les mots dans un ordre correct pour former une phrase, en conjuguant le verbe et en ajoutant la préposition qui convient.
Le choix du mode et du temps auxquels le verbe doit être conjugué est laissé à l'élève.
Exemple : "estar - mi tio - Espana"
qui peut devenir
"mi tio esta en Espana"

4.c. Verbes à l'infinitif avec mode et temps demandés.

L'intitulé de l'exercice serait :
"Formez des phrases avec les mots suivants. Placez la préposition et conjuguez le verbe à la forme appropriée".
L'élève doit ici réécrire les mots dans un ordre correct pour former une phrase, en conjuguant le verbe au mode et au temps demandés et en ajoutant la proposition adéquate.
Exemple : "estar - mi tio - Espana" (indicatif imparfait)
qui doit devenir
"mi tio estaba en Espana"

5. TYPE 5 : Formation de phrases.

Une liste de verbes est proposée à l'élève.
Ce dernier doit former une phrase incluant une préposition de lieu "a" ou "en" avec chaque verbe de la liste.
De cette proposition initiale de Monsieur Régimont, nous avons dérivé deux variantes.

5.a. Pas de temps demandé.

L'intitulé de cette première variante serait :
"Formez des phrases suivant le modèle : Sujet - Verbe - Préposition - Lieu. Employez un verbe de la liste".
Une liste de verbes à l'infinitif est proposée à l'élève. Celui-ci doit former une phrase, selon le modèle indiqué, pour chaque verbe de la liste. Ce dernier doit être conjugué correctement à un mode et à un temps choisis par l'élève.
Exemple : "Estar —> Mi tio esta en Paris"

5.b. Avec mode et temps demandés.

L'intitulé de cette seconde variante serait :
"Formez des phrases suivant le modèle : Sujet - Verbe - Préposition - Lieu. Conjuguez le verbe à la forme correcte".
Une liste de verbes à l'infinitif, accompagnés d'un temps et d'un mode demandés, est proposée à l'élève.
Celui-ci doit former une phrase selon le modèle indiqué, pour chaque verbe de la liste, et en conjuguant correctement ce dernier au mode et au temps demandés.
Exemple : "ir (futuro de indicativo) :
Los ninos iran al patio".

Tels sont les cinq types d'exercices que l'élève devra résoudre au cours d'une session de travail avec le didacticiel.
Il est bien entendu que tant les énoncés des différents exercices que les temps demandés seront rédigés en espagnol.

C H A P I T R E I I I .

INTRODUCTION AU DEVELOPPEMENT FUTUR DU DIDACTICIEL.

Dans le cadre de l'analyse de système, j'ai déterminé, en collaboration avec Monsieur Régimont, le scénario d'une session du didacticiel. Ce scénario repose sur l'enchaînement d'une série de grilles d'écran qui sont expliquées en détail à l'Annexe 1. Je vais simplement ici en esquisser les grandes lignes.

Il faut d'abord savoir que les idées développées dans ce chapitre n'ont pas été réalisées. Néanmoins, un chapitre leur est consacré et ce pour deux raisons importantes : d'une part, ces développements sont indispensables à la réalisation d'un didacticiel opérationnel ; d'autre part, elles représentent un investissement de temps de travail assez important. Les scénarii esquissés dans ce chapitre ont permis de découvrir les problèmes effectivement résolus, jusque et y compris la programmation, qui seront exposés dans la suite de ce mémoire.
Un contrôle d'accès au système est établi. Ce contrôle porte sur le nom de l'élève utilisateur. Le professeur introduira dans un fichier le nom des élèves pouvant utiliser le système et ce dernier vérifiera si le nom de la personne voulant commencer une session s'y trouve avant de l'autoriser à travailler. De plus, le professeur attribuera un niveau à chaque élève. Ce niveau déterminera le degré de difficulté des sessions d'exercices proposées. Par exemple, on peut imaginer qu'un élève de niveau 1 se voit proposer des exercices de types 1 et 2, un élève de niveau 2 des exercices de types 2 et 3, un élève de niveau 3 des exercices de types 3 et 4 et un élève de niveau 4 des exercices de types 4 et 5.
Mais cela n'est qu'une proposition et d'autres modalités concernant le contrôle d'accès, le niveau des élèves ou la gradation de la difficulté des sessions d'exercices peuvent être définies. Par exemple, on pourrait décider que le niveau des élèves soit modifié automatiquement par le système en fonction des résultats qu'ils ont obtenus. On pourrait également laisser ce soin au professeur qui déciderait lui-même et en fonction de ses propres critères de modifier le niveau de ses élèves.

La cote ou le score que l'élève a atteint est affichée en permanence dans le coin supérieur droit de l'écran. Elle est mise à jour après chaque modification. Les modalités de son calcul restent également à définir.
Au bas de chaque écran sont indiquées les commandes permettant à l'élève de passer à un autre écran, de signaler la fin de sa réponse ou de demander de l'aide.
Quand l'élève demande de l'aide, il se voit proposer un menu lui donnant à choisir entre :
- la liste des commandes disponibles
- la règle d'emploi des prépositions
- les règles de conjugaison
- les règles du système (durée, cotation, ...)

Chacun de ces quatre sujets particuliers est expliqué sur un ou plusieurs écrans. Au bas de ceux-ci sont indiquées les commandes permettant de faire défiler les écrans. Lorsque l'élève signale qu'il a fini de consulter une des règles, le système lui réaffiche l'écran d'où il était parti avant la demande d'aide.
Chaque demande d'aide concernant la règle d'emploi des prépositions ou les conjugaisons est sanctionnée par une diminution du score. On peut aussi imaginer un nombre maximum de consultation de règle alloué à chaque élève pour une session. Chaque demande d'aide serait répercutée dans la cotation et si le nombre maximum était atteint, le système mettrait prématurément fin à la session et donnerait une cote nulle. Une autre possibilité serait de sanctionner de plus en plus lourdement les demandes d'aide au fur et à mesure que leur nombre augmente.

Pour chaque type d'exercice, il existe un écran d'introduction et un écran d'exercice proprement dit.
L'écran introductif reprend l'intitulé de l'exercice et un exemple de résolution de celui-ci.
L'écran d'exercices reprend l'historique des réponses passées de l'élève, l'intitulé de l'exercice et la "phrase-question" courante. Au fur et à mesure du déroulement de la session, l'historique des réponses passées défile vers le haut. Pour les exercices de type 3, il existe également un écran d'exercices où l'historique des réponses est remplacé par la liste des verbes proposés pour résoudre l'exercice.

Quand l'élève donne une réponse incorrecte, l'écran de correction lui apparaît. Sur celui-ci s'afficheront les dialogues entre l'élève et le système permettant de corriger la réponse. Cela fait, l'écran d'exercices d'où on était parti réapparaît. Sur ce dernier, la bonne réponse obtenue à l'issue du processus de correction est affichée dans l'historique des réponses passées suivie d'un signe distinctif indiquant à l'élève que sa réponse à cette question n'était pas correcte.
Le système corrige une phrase de gauche à droite. S'il arrive au bout de la phrase sans s'arrêter, c'est qu'il n'a pas trouvé de faute. Dans le cas contraire, il s'arrête à la première faute qu'il rencontre et ne se préoccupe pas du reste de la phrase. Ainsi, s'il trouve une faute dans le premier mot, il n'ira pas plus loin dans la phrase et signalera tout de suite la faute à l'utilisateur.
Les causes de rejet d'une réponse par le système peuvent être les suivantes :

1. faute dans la préposition : peut apparaître dans tous les types d'exercice. Il faut aussi envisager le cas où la préposition "**A**" et l'article défini "**EL**" se contractent en "**AL**". Je n'ai pas traiter ce sujet de manière plus approfondie et il faudra en tenir compte dans l'écriture de la réponse par l'élève dans les exercices de type 1, 2 et 3
2. faute dans le verbe : peut apparaître dans tous les types d'exercices, sauf le type 1. Il peut y avoir des fautes d'orthographe dans le radical et des fautes de conjugaison dans la terminaison.
3. faute dans un groupe nom : peut apparaître dans les exercices de type 4 et 5. Il peut y avoir trois formes de groupe nom : article et nom, nom seul et pronom. Dans les exercices de type 4 et 5, l'élève doit écrire toute la phrase. Il risque donc de faire des fautes d'orthographe ou de frappe notamment dans le sujet et le lieu qui sont des groupes noms.
4. faute dans le choix du verbe : peut apparaître dans les exercices de type 3. Si le radical du verbe écrit par l'élève n'a aucun caractère qui soit à la bonne place par rapport à un verbe correct.
5. faute dans l'ordre des mots : peut apparaître dans les exercices de type 4 et 5. Les élèves doivent y écrire des phrases dans un ordre correct. Or, l'ordre des mots n'est pas très important en Espagnol. La seule chose qui importe dans le cas qui nous occupe est que la préposition soit juste devant le groupe nom

du complément de lieu. Il faut également que l'intégrité des groupes noms soit respectée, c'est-à-dire que l'article soit placé devant le nom auquel il se rapporte.

6. non respect des espaces entre mots : peut apparaître dans tous les types d'exercices. Dans les exercices de type 1, l'élève a la place nécessaire pour écrire les deux lettres nécessaires aux prépositions "A" ou "EL". Cet espace est entouré de chaque côté d'un 'blanc' pour le séparer respectivement du verbe et du groupe nom du lieu.
   Dans les exercices de type 2 et 3, l'élève doit écrire à la fois le verbe et la préposition. Il disposera donc de 2 positions pour la préposition, plus les 2 'blancs' séparant la préposition du reste de la phrase, et de X positions pour le verbe. X est un nombre au moins égal au nombre maximum de lettres d'un verbe conjugué employées par le système augmenté du nombre de 'blancs' séparant le verbe du reste de la phrase, c'est-à-dire 2. Dans le cas qui nous occupe, on peut sans craindre de surprise fixer X à 20.
   Dans les exercices de type 4 et 5, l'élève doit écrire toute la phrase. Pour cela, une ligne lui est allouée. S'il ne sépare pas 2 mots par un espace, on considère que les deux mots n'en forment plus qu'un ce qui occasionnera une faute d'orthographe sur le premier mot.

Le dialogue de correction entre le système et l'utilisateur se déroulera comme suit.
La phrase-question sera affichée en haut de l'écran. Sous celle-ci apparaîtra la réponse incorrecte de l'élève qui a déclenché le processus de correction. Dans cette dernière, le premier élément jugé incorrect à partir de la gauche est mis en évidence d'une façon à convenir : clignotement, changement de couleur, encadrement ... etc Cet élément est soit une lettre, soit un mot. De plus, sous cette réponse est affichée la raison du refus.
A ce moment, le système donne l'occasion à l'élève de corriger sa réponse. Pour ce faire, les modalités varient suivant le type d'exercice.
Dans les exercices de type 1, toute la phrase est affichée, exceptée la préposition puisque c'est la seule chose que l'élève doit écrire et donc le seul élément susceptible d'être incorrect ; l'élève doit donc simplement écrire la préposition.

Dans les exercices de type 2 et 3, seuls le verbe et la préposition doivent être écrits par l'élève. Ni l'un ni l'autre ne seront affichés et l'élève doit les réécrire.
Dans les exercices de type 4 et 5, l'élève doit de toutes façons réécrire toute la phrase.
L'élève a le droit de donner deux mauvaises réponses concernant le même élément (sujet, préposition, conjugaison, choix du verbe, ...) avant que le système ne commence à l'aider.
Ainsi, l'élève dans deux réponses successives fait une faute dans la préposition, soit qu'il ne choisisse pas la bonne préposition, soit qu'il l'orthographie mal. Alors le système lui corrigera une lettre dans sa réponse et l'invitera à donner une nouvelle réponse et ce jusqu'à ce qu'il trouve la réponse correcte.
Dans le cas de faute dans le verbe, faute d'orthographe ou de conjugaison, le système procédera de même après deux réponses incorrectes consécutives et proposera chaque fois une lettre correcte jusqu'à obtenir une réponse sans faute.
Pour une faute dans un groupe nom, le mécanisme est identique.
Dans le cas d'une faute dans l'ordre des mots, comme il a été dit plus haut, elle ne peut venir que d'un mauvais placement de la préposition.
Lorsque l'élève a utilisé ses deux droits à l'erreur sans avoir trouvé la bonne réponse, le système la lui donne en plaçant la préposition correctement.
Lorsqu'il y a une faute dans le choix du verbe dans les exercices de type 3, l'aide fournie par le système après deux erreurs se présentera soit sous la forme du radical du verbe pour les exercices du type 3 b, où il doit encore être conjugué, soit sous la forme du verbe tout entier pour les exercices de type 3 a, où il ne doit pas être conjugué.
Lorsqu'une réponse correcte a été obtenue comme il a été dit plus haut, l'écran de correction s'effacera pour faire place à l'écran d'exercices d'où on était parti.
Les modalités de passage d'un type d'exercices à un autre ne sont pas encore fixées. Néanmoins, plusieurs possibilités peuvent être envisagées. La plus simple serait le passage au type d'exercice suivant après une série de X questions, X étant un nombre fixé à l'avance et une fois pour toutes.
On peut également imaginer faire varier la valeur de X en fonction du niveau de l'élève. Une autre possibilité serait de lier le passage au type d'exercice supérieur au score atteint par l'élève.

Ainsi, si ce dernier n'a pas obtenu une cote de Y sur une série de X questions, il recommencerait une autre série de X questions jusqu'à atteindre ce score.
La cote peut être calculée uniquement sur la dernière série de questions, sur l'ensemble des séries de questions de l'exercice considéré ou sur l'ensemble des exercices de la session déjà faits.
On peut encore imaginer que l'élève passe au type d'exercice supérieur s'il a répondu sans faute à une série de X questions consécutives.
Le système lui repose des questions jusqu'à ce qu'il réalise une telle série.
L'élève pourrait aussi devoir répondre sans faute à toutes les questions. Celles auxquelles il n'aurait pas répondu correctement lui seraient reposées à la fin de la série de questions d'un type d'exercices jusqu'à ce qu'il ne fasse plus de fautes.
Comme on le voit, les possibilités sont nombreuses et il en existe certainement encore d'autres. Il reste à faire un choix et à le réaliser.

A la fin de la session apparaît un écran indiquant à l'élève sa cote finale ainsi que sa valeur relative par rapport à d'autres cotes réalisées par d'autres élèves et/ou lui-même. En fait, on pourrait imaginer une sorte de classement des sessions effectuées en fonction des scores atteints. Chaque session serait identifiée par le nom de l'élève, le niveau de difficulté de la session, le score atteint et éventuellement la date. Ce classement incorporerait la session qui se termine en la mettant en évidence et serait toujours affiché à la fin, comme écran de sortie du système.

C H A P I T R E I V.

ETUDE DE FAISABILITE DU DIDACTICIEL.

Arrivé à ce point de mon travail, la suite logique aurait été de continuer l'analyse du système pour terminer par la programmation et les tests. Mais avant cela, il fallait d'abord savoir si ce travail en valait la peine. En d'autres termes, il importait de savoir si le didacticiel était faisable en termes d'algorithmes, de place mémoire disponible et de vitesse d'exécution.
Deux points précis semblent poser problème : la correction des réponses proposées par l'élève et l'origine des phrases à proposer à l'élève.

1.Correction des réponses.

Lorsque la réponse de l'élève se limite à la préposition comme dans les exercices de type 1 ou à la préposition et au verbe comme dans les exercices de type 2 et 3, elle ne pose pas de problème de correction. Le système sait quelle est la réponse correcte et pourra sans mal vérifier la réponse de l'élève en la comparant avec celle dont il dispose.
Mais si la réponse est une phrase entière comme dans les exercices de type 4 et 5, sa correction pose des problèmes.
Dans les exercices de type 4,le système connaît la réponse correcte quand il pose la question et le mécanisme de correction est similaire à celui des exercices de types 1, 2 et 3 ; il compare le modèle de réponse correcte dont il dispose avec la réponse de l'élève.
Mais pour les exercices de type 5, le système ne connaît pas la réponse correcte à la question qu'il pose car il peut y en avoir plusieurs. Cet aspect de pluralité des réponses correctes est également présent pour les exercices de type 3, quand plusieurs verbes de la liste conviennent pour une même phrase. La correction des réponses aux exercices du type 5 devra se faire sur deux plans : syntaxique et sémantique.

Sur le plan syntaxique se pose principalement le problème de l'orthographe des mots.
Lorsqu'il n'y a qu'une seule réponse correcte possible, il n'y a pas de problème. Une simple comparaison entre la réponse

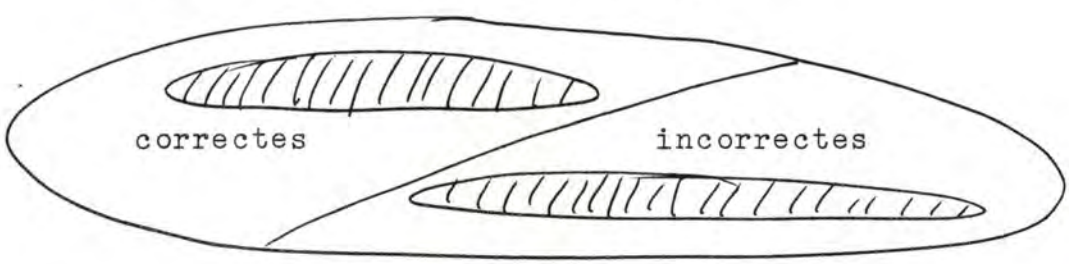
de l'élève et la réponse modèle attendue suffit.
Mais quand il y a plusieurs réponses correctes possibles, les problèmes surgissent. On pourrait peut-être imaginer comparer la réponse de l'élève avec toutes les réponses correctes possibles. Mais on rencontrerait alors de grandes difficultés dans l'établissement de la liste de celles-ci. De plus, la dimension de cette liste serait un inconvénient non négligeable et on courrait toujours le risque qu'elle ne soit pas exhaustive.
Et le fait que la réponse aux exercices de type 5 est composée de plusieurs éléments entraîne des difficultés dans la localisation de la faute en cas de réponse incorrecte.
Est-ce le premier mot,le sixième, le troisième, le dernier ?
Pour toutes ces raisons, la conception d'un module de correction automatique d'orthographe est indispensable.
L'objectif de ce module est double.
Tout d'abord, il doit déterminer si l'orthographe d'un mot est correcte ou incorrecte. Dans le cas d'une faute d'orthographe, il devrait aussi déterminer dans la mesure du possible la version orthographiquement correcte du mot qui lui est proposé, autrement dit le modèle de réponse correcte.
Une fois le problème de la correction de l'orthographe résolu, la vérification de la conjugaison ne devrait pas poser trop de problèmes et l'aspect syntaxique de la correction de la réponse serait réglé.

L'aspect sémantique de ce problème paraît à première vue plus compliqué. Tous les informaticiens savent que l'informatique bute sur l'interprétation sémantique des langages naturels. Le didacticiel a pourtant besoin d'un module de vérification sémantique des phrases écrites par l'élève.
Une phrase peut être décomposée comme suit :
<phrase> : : = <nom> <verbe conjugué> <préposition> <nom>
Si on suppose qu'il y a un algorithme qui s'occupe de la conjugaison, ce qui est tout à fait faisable, la phrase se ramène alors à
<phrase> : : = <nom> <verbe> <préposition> <nom>

Or, le vocabulaire communiqué par Monsieur Régimont et qui est celui dont disposent ses élèves compte 375 substantifs et pronoms, 15 verbes employant les prépositions de lieu "a" et/ou "en" et ces deux dernières. A priori, on pourrait donc générer 315 X 15 X 2 X 315 = 4.218.750 phrases. Parmi celles-ci, beaucoup sont incorrectes. Ce sont ces dernières que le module doit détecter pour les refuser. Deux options se situant complètement à l'opposé l'une de l'autre sont possibles. La moins élégante serait la construction d'un fichier de 4.218.750 entrées. Chacune de ces entrées indiquerait si la phrase qu'elle représente est sémantiquement correcte ou pas. Dans le même ordre d'idée, on pourrait également imaginer un fichier ne contenant que les phrases sémantiquement correctes parmi les 4.218.750 possibles à priori. Mais, outre son caractère peu élégant, cette option du fichier de phrases présente l'inconvénient de sa construction : soit il faut fournir le travail fastidieux et considérable d'écriture des 4.218.750 phrases à priori possibles et le tri entre les phrases sémantiquement correctes et les phrases incorrectes manuellement, soit il faut concevoir un algorithme exécutant ce travail. De toutes manières, il faudrait que toutes les phrases aient été générées au préalable.

L'autre option est beaucoup plus élégante. Elle implique la construction d'un modèle permettant au système de vérifier la sémantique des phrases qui lui sont soumises, sans que celles-ci aient été construites et stockées au préalable. Ces phrases répondraient aux critères de l'exercice de type 5 (sujet - verbe - préposition - complèment de lieu). C'est cette option qui a été choisie. L'inconvénient est que ce modèle n'est valable que pour certaines phrases et il restera toujours un ensemble de phrases sur la sémantique desquelles le modèle ne pourra se prononcer. Il faudra donc veiller à ce que cet ensemble soit le moins important mossible. L'alternative est résumée par ce schéma :

ensemble des phrases possibles à priori



- avec l'option minimum, le fichier, la frontière entre les phrases correctes et les phrases incorrectes est bien définie.
- avec l'autre option, la frontière n'est pas bien définie et on ne définit qu'une fraction (hachurée) de chaque partie.

2.Origine des phrases.

Pour l'origine des phrases à proposer à l'élève, deux possibilités viennent à l'esprit : soit elles sont contenues dans un fichier constitué au préalable, soit elles sont construites par programme. Ce problème est à rapprocher de la correction de la sémantique des phrases. On peut espérer qu'un système sachant distinguer une phrase sémantiquement correcte d'une phrase sémantiquement incorrecte puisse également générer des phrases sémantiquement acceptables.

Si on prend l'option de constituer un fichier de phrases, l'automatisation des séances de rattrapage devient limitée. En effet, il incombera au professeur d'inventer des phrases puis de les introduire dans le fichier afin de rendre le didacticiel opérationnel.
Ce travail est analogue au travail de génération des phrases possibles et sémantiquement correctes destinées à la correction. La différence réside dans le fait que toutes les phrases correctes possibles employant le vocabulaire connu doivent être générées pour la correction, alors qu'on peut envisager ne retenir qu'un sous-ensemble de celles-ci pour la génération des phrases servant de support aux exercices. Ce processus devrait même être recommencé périodiquement pour renouveler le corpus de phrases possibles. La préparation des exercices par le professeur serait quasi semblable à la confection de feuilles stencilées à l'intention des élèves. Seule la correction se ferait automatiquement, et encore serait-elle préparée. Pour le professeur, le gain de temps apporté par l'informatisation des exercices serait fortement diminué. Cette option devrait donc être abandonnée.

Par contre, l'idée d'un programme de génération de phrases permettrait d'automatiser jusqu'à la conception même des questions. Le système générerait la totalité des phrases sur lesquelles portent les exercices et la seule intervention de l'enseignant serait l'introduction des élèves pouvant utiliser le didacticiel, et le contrôle des résultats.
Il faut donc que les phrases générées par le système soient sémantiquement correctes. Pour ce faire, le module de génération de phrases utilisera le modèle de sémantique utilisé pour la correction. Les deux problèmes, la génération de phrases sémantiquement correctes et la correction sémantique de phrases, sont donc similaires quant à leur aspect sémantique et seront résolus grâce au même modèle.

La correction de l'orthographe et la génération et le contrôle sémantique des phrases sont donc les fondements de la faisabilité du système. S'ils sont faisables et exécutables sous certaines contraintes de place mémoire disponible et de vitesse d'exécution, le reste du système pourra être construit. Sinon il serait inutile de tenter de construire un système qui ne fonctionnerait pas. Il s'agit de savoir si ces modules sont réalisables et satisfaisants quant à leur place mémoire et à leur vitesse d'exécution. Mon but sera donc de les réaliser et de les tester. Les chapitres suivants explicitent la démarche suivie pour ce travail.

Ce mémoire est donc un travail préparatoire destiné à montrer la faisabilité de l'ensemble du didacticiel.

C H A P I T R E V .

LE MODULE DE CORRECTION D'ORTHOGRAPHE.

Introduction

Pendant toute la session et dans tous les types d'exercices, l'utilisateur est susceptible de commettre des fautes d'orthographe ou de frappe en écrivant sa réponse. Mais les exercices de type 5 sont les seuls où le système ne connait pas la ou les réponses correctes à l'avance. C'est pour ces exercices que le présent module de correction automatique d'orthographe a été développé et réalisé.

Conçu en collaboration avec David Gouthière, ce module n'est pas destiné à corriger des fautes d'orthographe seulement en espagnol mais aussi dans n'importe quelle autre langue. En effet, David avait besoin d'un système corrigeant les fautes d'orthographe ou de frappe éventuellement présentes dans des phrases inventées et écrites en français par l'utilisateur de son système.

Son problème était donc analogue à celui posé par les réponses aux exercices du type 5 du présent système. Par conséquent, Monsieur Cherton nous a suggéré de collaborer l'un avec l'autre et de réaliser à nous deux le module décrit dans les pages qui suivent.

Travaillant l'un sur des mots français, l'autre sur des mots espagnols, notre algorithme ne pouvait employer aucune particularité de l'une de ces deux langues, ni d'ailleurs de n'importe quelle autre. Il a néanmoins été développé et testé, surtout en ce qui concerne sa vitesse et la justesse de ses corrections, sur un vocabulaire espagnol. En effet, nous disposions de la liste des substantifs espagnols enseignés par Monsieur Régimont à ses élèves et donc susceptibles d'être employés par le didacticiel d'espagnol, alors que le vocabulaire français du système développé par David n'était pas encore défini.

I. Présentation du problème.

Le module de correction automatique d'orthographe ne doit pas simplement dire si le mot qu'on lui soumet est bien orthographié. Il doit également, dans le cas où il y a une faute d'orthographe, proposer une correction, un mot qui existe vraiment (le mot mal orthographié n'existe pas, ce n'est pas un mot), qui ressemble le plus à ce qui a été soumis à la correction et que James L. Peterson (1) appelle un signe (token).

Donc, le module reçoit un signe qui est défini comme une suite de caractères commençant par un espace et se terminant par un autre espace. Le module doit déterminer si le signe est un mot, c'est-à-dire s'il est correctement orthographié. Si ce n'est pas le cas, il doit proposer le mot qui ressemble le plus au signe. S'il ne trouve pas de mot ressemblant assez au signe, il dit qu'il ne connait pas le signe. Cela signifie soit que le signe est un mot incorrectement orthographié mais comportant trop de fautes pour être corrigé, soit que le signe est un mot qui n'appartient pas au vocabulaire dont dispose le système.

La contrainte principale à laquelle doit se soumettre le module est la limitation du temps de réponse à 2 ou 3 secondes maximum. En effet, le système détectera et tentera de corriger les fautes d'orthographe de toute une phrase écrite par l'élève dans le cadre des exercices de type 5. Cette phrase sera constituée d'environ 6 mots. Le module de correction s'exécutant sur chaque mot, un temps de réponse d'une dizaine de secondes maximum pour une phrase constitue une limite à ne pas dépasser. Ce délai peut également jouer un rôle de dissuasion pour encourager l'élève à ne plus commettre de fautes. En effet, si le temps de réponse est insignifiant lorsque le mot est correctement orthographié, il peut être de 3 ou 4 secondes dans le cas contraire. Donc, plus la phrase comporte de mots incorrectement orthographiés, plus le temps de réponse sera important, les temps d'exécution du module de correction d'orthographe sur chaque mot s'additionnant. L'élève s'apercevant que le délai entre sa réponse et la correction du système augmente avec le nombre de fautes qu'il commet, on espère qu'il fera plus attention à l'exactitude de ses réponses.

Une des limites les plus contraignantes que rencontrent les correcteurs d'orthographe est la taille du dictionnaire. Nous en parleront d'ailleurs plus loin, au paragraphe II du présent chapitre. D'habitude, l'importance de cette taille pose des problèmes assez importants de stockage en mémoire et de rapidité d'accès à un élément de ce dictionnaire. Mais, dans notre cas, la taille du dictionnaire est réduite (2.193 caractères répartis en 316 mots pour le vocabulaire de test) ce qui représente un gros avantage pour le stockage, la rapidité d'accès et la recherche d'un élément.

II. Base bibliographique.

Introduction

Les recherches bibliographiques que nous avons menées nous ont permis, entre autres, de découvrir le livre de James L. Peterson, 'Computer Programs for Spelling Correction : An Experiment in Program Design'(1). L'auteur y résume les divers travaux concernant la détection et la correction automatiques des fautes d'orthographe par ordinateurs et montre comment ils peuvent être appliqués à la création d'un véritable correcteur d'orthographe d'intérêt général. Cette partie a aussi fait l'objet d'un article dans 'Communications of A.C.M.' en décembre 1980 (20). Dans la deuxième partie de son ouvrage, la plus importante, il démontre l'application de principes modernes d'architecture de programme sur un programme de correction d'orthographe. Enfin, dans la troisième et dernière partie, l'architecture du programme est convertie en Pascal, compilée et exécutée sur un DEC-20 du 'Massachusetts Institute of Technology.'

La première partie de l'ouvrage a évidemment attiré principalement notre attention. Les résultats de la recherche bibliographique qui y est détaillée nous ont fourni les principes généraux sur lesquels est basé notre programme. De plus, elle nous a permis de mieux cerner le problème en l'introduisant dans un contexte plus général et en présentant les solutions qui ont été étudiées auparavant. Le paragraphe qui va suivre n'est d'ailleurs qu'un bref résumé de cette partie de l'ouvrage de James L. Peterson.

II. 1. Présentation du problème

Il y a deux types de programmes d'orthographe : les vérificateurs d'orthographe et les correcteurs d'orthographe. Les vérificateurs d'orthographe identifieront simplement les mots mal orthographiés. Les correcteurs d'orthographe détectent les mots mal orthographiés et essaient de déterminer le mot bien orthographié le plus probable qu'ils représentent. Le module présenté ici est donc un correcteur d'orthographe.

Les trois principales sources d'erreur sont les suivantes :

- ignorance par l'auteur de l'orthographe correcte : probablement liée à la différence entre la phonétique et l'orthographe du mot.
- faute de frappe : probablement liée à la position des touches sur le clavier et à des erreurs spécifiques dans le mouvement des doigts.
- erreurs de transmission et de stockage à l'intérieur du système.

Nous ne nous préoccuperons que des deux premières sources. La première motivation de la recherche sur le problème des fautes d'orthographe était la correction d'erreurs à l'entrée de données. Par exemple, Davidson (2) s'est intéressé à la recherche de noms potentiellement mal orthographiés dans des listes de passagers d'avion. Carlson (3) s'est intéressé aux noms et places dans une base de données généalogiques. Freeman (4) a travaillé avec des noms variables et des mots clés dans le langage de programmation CORC alors que Mac Elwain et Evans (5) se sont intéressés à l'amélioration des résultats d'un système destiné à reconnaître le code morse. Chacun de ces projets considérait le problème de l'orthographe comme un aspect seulement d'un problème plus vaste, et non comme un outil logiciel séparé.

De même, plusieurs études ont été publiées concernant le problème général d'algorithmes de vérification et de correction de chaînes : Damerau en 1964 (6), Alberga en 1967 (7),Riseman et Ehrich en 1971, Wagner et Fischer en 1974 (9), Riseman et Hanson en 1974 (10) et Lowrance et Wagner en 1975 (11).

Les recherches concernant la correction d'orthographe remontent à 1957 mais le premier vérificateur d'orthographe écrit comme un programme d'application et non comme un objet d'expérimentation et de recherche semble être SPELL pour le DEC-10, écrit par Ralph Gorin à Stanford en 1971.
Ce programme et ses dérivés ont été et sont toujours largement employés sur les systèmes DEC-10 et DEC-20. D'autre part, le système d'exploitation UNIX fournit deux vérificateurs d'orthographe : TYPO et SPELL. Enfin, un autre vérificateur d'orthographe a été écrit pour les systèmes IBM/360 et IBM/370.
Tel était l'état des travaux sur le sujet en 1980 ; l'article de J.L Peterson citant toutes ces sources ayant paru dans les Communications of A.C.M. de décembre 1980. Entre cette date et octobre 1983, moment où le développement du module a commencé, il semble qu'il n'y ait rien eu d'intéressant. En effet, nos recherches dans les 'Communications of A.C.M.' et les 'Computer and Control Abstracts', ainsi que dans les 'Computer Abstracts', n'ont rien donné. Néanmoins, certains articles ou ouvrages, non référencés dans les revues que nous avons consultées ou indisponibles à la bibliothèque Moretus Plantin ont pu échapper à nos recherches.

II. 2. Premier stade : établissement d'une liste de signes.

Afin de détecter les fautes d'orthographe dans un texte, il faut d'abord dresser une liste de tous les signes distincts de celui-ci. Pour ce faire, il faut prendre une décision sur ce qu'est un mot potentiel, aussi appelé signe. Un mot est une suite de lettres séparées par des délimiteurs. Ceux-ci sont des blancs ou des caractères spéciaux tels que virgules, points, colonnes ou d'autres encore ...
Dans la plupart des cas, la classification d'un caractère comme lettre ou délimiteur est claire, mais il faut se préoccuper particulièrement de l'interprétation des nombres, trait d'union et apostrophes.
Il faut également prendre en compte la différence entre minuscules et majuscules. La plupart du temps, les lettres sont minuscules bien que la première lettre soit majuscule au début de la phrase, ou lorsqu'il s'agit d'un nom propre. Nous n'avons pas prêté attention à ce problème car la machine sur laquelle nous avons travaillé ne disposait que des majuscules. Mais, le cas échéant, les modifications à apporter seraient relativement minimes.

Le mot placé en début de phrase est facilement détectable et il suffit de vérifier si sa première lettre est majuscule avant de la transformer en minuscule pour le reste du traitement. De même, les noms propres commençant par une majuscule et la représentation des majuscules et des minuscules étant différentes en mémoire, ils se traitent comme n'importe quel autre mot. Pour le correcteur d'orthographe, "Madrid" et "madrid" sont deux mots différents puisque "M" et "m" sont différents.

Le problème de constitution d'une liste de signes s'est trouvé simplifié dans notre travail. En effet, le module que nous avons développé reçoit en entrée une chaîne de caractères (qui peut comprendre un ou plusieurs blancs) et qu'il considère comme un mot. Il n'a donc pas besoin de différencier les signes qu'on lui soumet en entrée. Ce travail devra néanmoins être effectué par un autre module.

II. 3. Utilisation de fréquences de digrammes et trigrammes.

II. 3. 1. Introduction : les digrammes et trigrammes.

Une paire de lettres est appelée digramme et un triplet de lettres trigramme. S'il y a 28 lettres (alphabet, blanc et apostrophe), alors il y a 28^2 (784) digrammes et 28^3 (21.952) trigrammes. Leur fréquence est très variable, beaucoup d'entre eux étant très rares. Une étude a découvert que dans un large échantillon de textes, on ne retrouvait que 550 digrammes (70%) et 5000 trigrammes (25%) différents. Si un signe contient un ou plusieurs digrammes ou trigrammes très rares, il est probablement mal orthographié et la faute ou les fautes sont presque certainement localisées dans ceux-ci.

L'utilisation de digrammes et trigrammes pour détecter les fautes d'orthographe et proposer les corrections constitue une des techniques les plus répandues dans la littérature. Outre dans les deux ouvrages déjà cités ci-dessus, on en parle également dans des articles de Harmon en 1962 (14), Mac Elwain et Evans en 1962 également (5), Vossler et Branston en 1964 (15), Carlson en 1966 (3), Cornew en 1968 (16), Riseman et Ehrich en 1971 (8) et Riseman et Hanson en 1974 (10).

II. 3. 2. Une méthode statistique de correction d'orthographe.

Je m'attarderai plus particulièrement sur l'article de Cornew qui est un des seuls que nous avons pu consulter à la bibliothèque Moretus Plantin.

II. 3. 2. 1. Le principe.

Dans la technique que Cornew décrit et qu'il applique à l'anglais, on stocke les 27^2 (26 lettres de l'alphabet + 1 espace) fréquences de digramme de la **langue** anglaise dans une table. Ces fréquences sont le nombre d'occurence d'1 digramme sur 10.000. La table est une matrice à deux dimensions dont l'indice ligne est la première lettre d'un digramme et l'indice colonne la seconde lettre.

LETTER OF THE SECOND POSITION

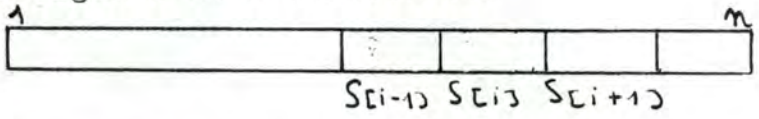
LETTER OF THE FIRST POSITION	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	SPACE
A	1	32	39	15		10	18		16		10	77	18	172	2	21	1	101	67	124	12	24	7		27	1	49
B	8				58				6	2		21	1		11			6	5		25				19		2
C	44		12		55	1		46	15		8	16			59	1		7	1	38	16		1				7
D	45	18	4	10	39	12	2	3	57	1		7	9	5	37	7	1	10	32	39	8	4	9		6		222
E	131	11	64	107	39	23	20	15	40	1	2	46	43	120	45	32	14	154	145	80	7	16	41	17	17		446
F	21	2	9	1	25	14	1	6	21	1		10	3	2	38	3		4	8	42	11	1	4		1		99
G	11	2	1	1	32	3	1	16	10			4	1	3	23	1		21	7	13	8		2		1		50
H	84	1	2	1	251	2		5	72			3	1	2	46	1		8	3	22	2		7		1		68
I	18	7	55	16	37	27	10				8	39	32	169	63	3		21	106	88		14	1	1		4	2
J					2										4						4						
K					28				8					3	3				2	1			3		3		17
L	34	7	8	28	72	5	1		57	1	3	55	4	1	28	2	2	2	12	19	8	2	5		47		49
M	56	9	1	2	48			1	26				5	3	28	16			6	6	13		2		3		40
N	54	7	31	118	64	8	75	9	37	3	3	10	7	9	65	7		5	51	110	12	4	15	1	14		135
O	9	18	18	16	3	94	3	3	13		5	17	44	145	23	29		113	37	53	96	13	36		4	2	83
P	21	1			40			7	8			29			28	26		42	3	14	7		1		2		13
Q																					20						
R	57	4	14	16	148	6	6	3	77	1	11	12	15	12	54	8		18	39	63	6	6	10		17		95
S	75	13	21	6	84	13	6	36	42		2	6	14	19	71	24	2	6	41	121	30	2	27		4		274
T	56	14	6	9	94	5	1	315	128			12	14	8	111	8		30	32	53	22	4	16		21		196
U	18	5	17	11	11	1	12	2	5			28	9	33	2	17		49	42	45				1	1	1	5
V	15				53				19						6												
W	32		3	4	30	1		48	37			4	1	10	17	2		1	3	6	1	1	2				7
X	3		5		1				4						1	4				1	1						3
Y	11	11	10	4	12	3	5	5	18			6	4	3	28	7		5	17	21	1	3	14				132
Z					5				2			1														1	
SPACE	247	85	94	91	55	84	25	95	118	14	4	45	79	43	142	82	1	62	138	363	28	8	127	1	15	1	

matrice de digrammes

En consultant cette matrice, on constate que le digramme "ON" revient 145 fois sur 10.000,le digramme "TH" 315 fois sur 10.000, que les digrammes "KA", "KB", "KC", "KD", etc... n'apparaissent jamais sur 10.000 observations, que le seul digramme dont "Q" est la première composante est "QU", etc...

Dans le cas où le signe donné ne se trouve pas dans le dictionnaire, ce signe est donc un mot incorrectement orthographié. On utilise la table des fréquences de digrammes pour examiner les voisins à gauche et à droite de chaque lettre dans le signe afin de trouver la lettre la moins probablement correcte. Dans cette recherche, chaque lettre est membre de deux digrammes dont les fréquences sont multipliées et associées à cette lettre commune.

- soit le signe S de n caractères.



- soit le digramme (S [i - 1] , S [i]) dont la fréquence est $F_{i-1,\,i}$
- soit le digramme (S [i] , S [i + 1]) dont la fréquence est $F_{i,\,i+1}$

Le produit $P_i = F_{i-1,\,i} \times F_{i,\,i+1}$ représente la fréquence du caractère S_i entouré à gauche de S [i - 1] et à droite de S [i] .

La lettre à laquelle est associé le produit minimum peut alors être considérée comme celle qui est la moins probablement correcte à cette place dans le mot. On considère donc que la probabilité d'exactitude des lettres croît avec le produit des fréquences des digrammes auxquels elles participent. Celle dont le produit des fréquences des digrammes est maximum est par conséquent considérée comme la plus probablement correcte à cette place dans le mot.

Pour essayer de corriger ce mot, on essaie de remplacer ses lettres, une après l'autre, jusqu'à ce qu'on ait épuisé toutes les possibilités en lettres à remplacer et en lettres remplaçantes ou qu'on ait ainsi formé un mot se trouvant dans le dictionnaire. Ces remplacements se font dans l'ordre croissant des produits des fréquences des digrammes.

Les lettres remplaçantes sont choisies selon les modalités suivantes. Ayant déterminé la lettre à remplacer, on détermine ses deux voisines de gauche et de droite dans le mot. On consulte alors dans la table des fréquences des digrammes la ligne dont l'indice est la lettre voisine à gauche de la lettre à remplacer et la colonne dont l'indice est la lettre voisine à droite. On fait alors le produit de chaque élément de cette ligne avec l'élément correspondant de cette colonne pour obtenir le produit des fréquences des digrammes que forment toutes les lettres de l'alphabet avec les mêmes lettres à gauche et à droite que la lettre à remplacer, $S_{[i-1]}$ et $S_{[i+1]}$.

- soit M : la matrice des fréquences de digrammes.

$M[S_{[i-1]}, *]$: le vecteur des fréquences des digrammes ayant la lettre $S_{[i-1]}$ comme première composante.

$M[*, S_{[i+1]}]$: le vecteur des fréquences des digrammes ayant la lettre $S_{[i+1]}$ comme seconde composante.

$P_* = M[S_{[i-1]}, *] \times M[*, S_{[i+1]}]$: le vecteur des produits des fréquences des digrammes pour les caractères situés entre $S_{[i-1]}$ et $S_{[i+1]}$.

La lettre dont le produit des fréquences des digrammes est maximum est considérée comme la plus probable à cette place dans le mot. On considère donc comme il a été dit plus haut que la probabilité d'exactitude des lettres décroît avec le produit des fréquences des digrammes auxquels elles participent.

On remplace donc la lettre choisie dans le mot par les lettres choisies selon les modalités ci-dessus dans l'ordre décroissant de leur produit des fréquences des digrammes, jusqu'à ce qu'on ait ainsi formé un mot se trouvant dans le dictionnaire, ou qu'on ait épuisé toutes les lettres remplaçantes qui peuvent être 25 au maximum (26 - la lettre à remplacer). Dans ce dernier cas, on choisit une autre lettre à remplacer selon les modalités décrites plus haut et on recommence le même processus jusqu'à ce qu'on ait trouvé un mot ou qu'on ait épuisé toutes les possibilités de remplacement.

II. 3. 2. 2. Exemple.

Voici l'exemple d'application de cette technique donné par Cornew.

signe	G	G	E	A	T
fréquence de digramme	25	1	32	131	124	196
produit de fréquences de digrammes	25	32	4192	16244	24304	

Le produit des fréquences des digrammes est minimum (25) pour le premier "G". Celui-ci est donc considéré comme le moins probablement correct. On essaie alors de le remplacer en choisissant dans le tableau des fréquences des digrammes la lettre dont le produit des fréquences des digrammes qu'elle forme avec "espace" à gauche et "G" à droite est maximum, et ainsi de suite en prenant chaque fois le produit inférieur ...

Corrections à la position la plus probable :

AGEAT	NGEAT	IGEAT	EGEAT	SGEAT	OGEAT	RGEAT
TGEAT	UGEAT	FGEAT	DGEAT	YGEAT	LGEAT	GGEAT
ZGEAT	XGEAT	WGEAT	VGEAT	QGEAT	PGEAT	MGEAT
KGEAT	JGEAT	HGEAT	CGEAT	BGEAT		

On a fait 26 remplacements (y compris G par G) sans résultat. On corrige alors la lettre dont le produit des fréquences des digrammes est juste supérieur c'est-à-dire 32 pour le deuxième G. On essaie de le remplacer en choisissant dans le tableau des fréquences des digrammes la lettre dont le produit des fréquences des digrammes qu'elle forme avec G à gauche et E à droite est maximum, et ainsi de suite en prenant chaque fois le produit inférieur ...

Corrections à la position la plus probable suivante :

GHEAT (GREAT) GEEAT GTEAT

On obtient au deuxième essai la forme orthographiquement correcte du mot.

II. 4. Utilisation d'un dictionnaire.

L'utilisation d'un dictionnaire de mots correctement orthographiés est un élément très important de la vérification d'orthographe. En effet, on vérifie si les signes en entrée sont dans le dictionnaire. Si oui, ce sont des mots correctement orthographiés. Si non, on sait qu'ils comprennent des fautes d'orthographe.

Si la liste de signes en entrée et le dictionnaire sont triés, un seul passage à travers les deux est nécessaire pour vérifier tous les signes, ce qui augmente la vitesse d'exécution.

Je ne m'étendrai pas plus longuement sur le sujet, Peterson parlant de constitution de dictionnaire à utiliser en anglais. Nous espérions que notre vocabulaire était relativement restreint (315 substantifs en espagnol) et qu'il ne nous poserait pas de problème de place et de vitesse de recherche. L'expérience nous a donné raison.

II. 5. Vérificateurs d'orthographe interactifs.

De tels programmes demandent à l'utilisateur ce qu'ils doivent faire des signes qu'ils ne retrouvent pas dans le dictionnaire. Ceux-ci sont soit mal orthographiés, soit inconnus du système. Le programme SPELL du DEC-10 s'exécute selon cette approche.

II. 5. 1. Modes d'opérations.

Lorsque le signe n'est pas dans le dictionnaire, quatre options sont possibles :

- remplacer : le signe inconnu est considéré comme inconnu et doit être remplacé par un mot correctement orthographié qu'on demande à l'utilisateur d'introduire.

- remplacer et se rappeler : le signe inconnu doit être remplacé par un mot spécifié par l'utilisateur et toutes les occurences futures de ce signe doivent aussi être remplacées par le nouveau mot.

- accepter : le signe est considéré comme correct dans ce contexte.

- accepter et se rappeler : le signe est considéré comme correct et toutes les occurences futures du signe sont correctes dans le contexte de ce même texte.

- édition : introduit un sous-mode d'édition permettant de modifier arbitrairement le fichier (dictionnaire) dans le contexte du texte.

II. 5. 2. Le dictionnaire.

Il existe plusieurs techniques permettant d'améliorer les performances et le temps de réponse de la recherche dans le dictionnaire, aspects importants dans un programme interactif et qui retiendront notre attention.

La structure du dictionnaire doit permettre des recherches très rapides. Elle doit être déterminée suivant la configuration du système employé telle que dimension de la mémoire, méthodes d'accès aux fichiers, ou existence d'une mémoire virtuelle. Si la mémoire est assez grande, on peut y charger tout le dictionnaire ce qui facilite et accélère considérablement les recherches. Si cette mémoire est virtuelle, la structure du dictionnaire devrait être telle que les défauts de page seraient minimes pendant la recherche. Si la mémoire est trop petite, une structure à deux niveaux est nécessaire. Celle-ci garde les mots les plus souvent utilisés en mémoire et accède aux autres sur disques quand c'est nécessaire.

Peterson cite également d'autres techniques qui tirent le plus souvent parti des particularités de la langue anglaise et s'appliquent à des dictionnaires de taille beaucoup plus grande que ceux que nous avons envisagés. Ainsi le Brown Corpus cité par Kucera et Francis en 1967 (17) contient 1.014.232 signes au total avec 50.406 mots distincts. Le mot le plus long compte 44 caractères et 99% des mots en comptent moins de 18. La longueur moyenne des mots est 8,1 caractères. Cela nous éloigne très fort de nos 315 mots dont les deux plus longs comptent 13 caractères et dont la longueur moyenne est 5,8 caractères.

II. 6. Correction des fautes d'orthographe.

II. 6. 1. Introduction.

Jusqu'ici, on se contentait la plupart du temps de détecter les fautes d'orthographe. A partir de maintenant, on va tenter de les corriger.

Le premier stade de correction est l'utilisation du mode "remplacer et se rappeler" du vérificateur d'orthographe interactif, telle qu'elle est expliquée plus haut.

Une approche similaire est d'incorporer au dictionnaire des mots couramment mal orthographiés dans cette forme avec l'indication de leur orthographe correcte. Par exemple, "language" serait une forme incorrecte courante de "langage" et "ortografe" une forme incorrecte de "orthographe". Mais cette technique est difficilement employable parce qu'on n'a pas de liste de fautes courantes connues et celles-ci paraissent quand même avoir une fréquence relativement faible.

II. 6. 2. Causes des fautes d'orthographe.

D'après Damerau (18), 80% des fautes d'orthographe sont le résultat de - la transposition de deux lettres;
- l'écriture d'une lettre de trop;
- l'oubli d'une lettre;
- l'écriture d'une lettre au lieu d'une autre.

II. 6. 3. Correcteur d'orthographe du DEC-10.

Les quatre règles sont à la base de la correction réalisée par le correcteur du DEC-10.

Pour chaque signe qu'on ne trouve pas dans le dictionnaire, construire une liste de tous les mots qui pourraient produire ce signe par une des quatre règles de Damerau. Ce sont les candidats à la bonne orthographe du signe.

Si la liste compte un seul candidat, demander à l'utilisateur si ce mot est l'orthographe correcte.

Si la liste compte plusieurs mots, le dire à l'utilisateur. Celui-ci peut demander à voir la liste pour choisir la bonne orthographe ou pour indiquer une des options : "remplacer", "remplacer et se rappeler", "accepter et se rappeler" ou "édition".

Par exemple, pour le signe ORD, on obtient les mots suivants : - transposition de deux lettres : ROD;
- suppression d'une lettre : OR;
- modification d'une lettre : ORB, ORE, OLD, ODD;
- ajout d'une lettre : CORD, FORD, LORD, WORD.

Ces dix mots sont candidats à la bonne orthographe de ORD.

II. 6. 4. Indice d'assortiment d'Alberga.

En 1967, Alberga (7) a suggéré de déterminer les longueurs de sous-chaînes communes entre un signe et des mots du dictionnaire. Ces longueurs sont utilisées pour déterminer un indice d'assortiment. Ce dernier peut être considéré comme la probabilité qu'un signe soit une forme mal orthographiée d'un mot du dictionnaire. Ainsi, le mot ayant la plus grande probabilité peut être considéré comme l'orthographe correcte la plus probable. L'inconvénient est que cet algorithme a un temps d'exécution assez long.

Wagner et Fisher en 1974 (9) et Lowrance et Wagner en 1975 (11) ont étudié des algorithmes semblables.

II. 6. 5. Utilisation des causes possibles d'erreurs pour augmenter la vitesse de correction.

L'emploi de certaines informations, notamment les sources d'erreurs possibles, peut souvent augmenter la vitesse moyenne de correction.

II. 6. 5. 1. Matrice de confusion.

Par exemple, les erreurs introduites par des instruments de saisie OCR (OPTICAL CODE RECOGNITION) sont seulement des substitutions d'une lettre par une autre. En effet, des caractères ne sont jamais insérés ou détruits par un lecteur OCR.
Il est alors possible de créer une matrice de confusion donnant la probabilité P_{ij} de lire un caractère i quand le caractère correct est j. Cette matrice peut être utilisée pour indiquer quels caractères devraient être pris en compte quand une lettre entrée doit être corrigée.

II. 6. 5. 2. Fréquence de digrammes et trigrammes.

Dans des systèmes plus généraux, les fréquences de digrammes et trigrammes pourraient être utilisées pour limiter le nombre d'orthographes alternatives à examiner (voir méthode de Cornew ci-dessus).

II. 6. 5. 3. Disposition de clavier.

L'agencement d'un clavier standard de machine à écrire peut également être utile dans la correction de fautes d'orthographe d'un texte introduit par ce moyen dans un système.
Cette idée est semblable à celle de la matrice de confusion (voir II. 6. 5. 1.).

II. 6. 5. 4. La phonétique des mots.

Une autre technique tente de résoudre le problème des mots dont l'auteur ne connaît pas l'orthographe correcte. Les fautes sont dans ce cas souvent basées sur les sons.
Il s'agit alors de convertir les signes dans une orthographe phonétique standard qui pourrait être utilisée pour trouver des mots phonétiquement similaires dans le dictionnaire.
Cette technique permet de corriger des erreurs telles que "f" pour "ph" ou "k" pour "qu".

II. 7. Utilisation d'affixes.

Le plus gros problème rencontré par les systèmes de correction d'orthographe est la taille excessive du dictionnaire. On peut réduire celle-ci en utilisant des préfixes ou suffixes communs à plusieurs mots, autrement appelés affixes.
Il suffit de retirer les affixes des mots pour obtenir leur racine que l'on stocke dans le dictionnaire.

Il existe deux stratégies possibles.

La plus simple consiste à comparer chaque signe avec une liste d'affixes courants et d'en retirer les affixes qu'on trouverait dans la liste. Par exemple, tous les signes se terminant par "s" verraient leur "s" final enlevé. Ensuite, le signe ainsi modifié, ou plutôt la racine du signe est comparée à chaque élément du dictionnaire. Si elle s'y trouve, le signe est considéré comme correctement orthographié.

Le gros inconvénient de cette approche est qu'elle ne détecte pas les signes mal orthographiés qui sont le résultat de la mauvaise concaténation de racines et d'affixes qui ne s'appliquent pas à celles-ci. Les signes "implyed" ou "fixs" sont des exemples de ce problème.

Pour éviter cela, il faut marquer chaque mot du dictionnaire avec ses affixes corrects.
Le procédé consiste d'abord à rechercher le signe dans le dictionnaire. S'il ne s'y trouve pas, on lui enlève ses affixes et on cherche si la racine ainsi obtenue se trouve dans le dictionnaire. Si oui, on vérifie si les affixes enlevés sont bien admis pour cette racine.

L'utilisation de cette technique est assez complexe. Elle est utilisée dans le programme SPELL du DEC-10.

Enfin, dans le même ordre d'idée, se place l'algorithme que J.L Dolby a développé en 1970 (19). Dans cet article, l'auteur passe en revue plusieurs combinaisons de vérifications de noms propres anglais. Il propose également un programme qu'il a testé avec succès sur des classes d'équivalence de noms propres d'un bottin téléphonique. L'idée est peut-être bonne.
Selon Peterson (1), l'utilisation d'affixes devrait être plus facile et plus nécessaire pour les langues romanes que pour l'anglais.

Malheureusement, les articles dont nous avons pu prendre connaissance traitaient tous de l'anglais et étaient trop particuliers à cette langue pour être généralisés au français et à l'espagnol. D'ailleurs si cela avait été, nous aurions dû concevoir deux algorithmes différents, ce que nous voulions éviter. En effet, notre algorithme doit être utilisé dans un didacticiel et, pour les raisons exposées dans le premier paragraphe de l'introduction, être compréhensible par des professeurs non informaticiens. Or un algorithme tenant compte des particularités d'une langue aurait été plus complexe et aurait entraîné des difficultés supplémentaires de compréhension chez les enseignants. Nous avons donc préféré concevoir un seul algorithme le plus simple possible pour être compréhensible par la majorité des enseignants et également le plus général possible pour qu'il soit utilisable dans plusieurs didacticiels.
Ces deux objectifs sont cependant soumis aux contraintes de place mémoire occupée et de temps d'exécution déjà citées plus haut. Mais, en définitive, il semble que l'algorithme que nous proposons satisfait tous ces objectifs.

III. Idée de solution.

III. 1. Vérification de l'orthographe d'un mot.

Contrairement aux algorithmes cités par Peterson, notre module ne travaille pas sur un texte mais sur un mot. Recevant une chaîne de caractères, il va vérifier son orthographe et si nécessaire tenter de la corriger. La vérification de l'orthographe suppose l'emploi d'un dictionnaire. Celui employé pour nos tests sera le vocabulaire espagnol fourni par Monsieur Régimont.
Le premier algorithme que nous avons écrit pour le traitement d'un mot est le suivant :

- lire le mot;
- le rechercher dans le dictionnaire;
- s'il est dans le dictionnaire, alors le mot est correctement orthographié; sinon le mot est incorrectement orthographié --> appel à traitement de mot incorrectement orthographié.

Nous prenons donc l'option qu'un mot ne se trouvant pas dans le dictionnaire est mal orthographié. Il incombera au traitement de mot incorrectement orthographié de tenter de le transformer en un mot se trouvant dans le dictionnaire.
Au départ, nous avons considéré que ce dernier contenait tous les mots dont le système tenait compte et sous toutes leurs formes. En effet, les différences grammaticales existant entre le français et l'espagnol nous interdisaient de concevoir un algorithme standard permettant de traiter des formes conjuguées et/ou accordées. Ce travail devant être effectué par après pour résoudre nos problèmes spécifiques, j'en parlerai plus loin. Pour le moment, nous prendrons l'hypothèse que le fichier renfermant le dictionnaire possède une entrée pour chaque mot.

III. 2. Traitement d'un mot incorrectement orthographié.

Pour tenter de corriger un mot, deux techniques ont attiré notre attention.

III. 2. 1. Distance entre deux mots.

Cette technique est dérivée de celle présentée par Wagner et Fischer (9) et Lowrance et Wagner (10). Elle associe une distance nulle si les mots sont identiques; et plus les mots sont différents, plus les distances augmentent.

III. 2. 1. 1. Justification du calcul de la distance.

Comme je l'ai déjà signalé, l'idée de cette technique vient de l'article de Wagner et Fischer, "The String - to - String Correction Problem" paru en 1974 (9). Les auteurs y présentent un algorithme déterminant la ressemblance entre deux chaînes de caractères comme le nombre minimum d'opérations nécessaires pour rendre une chaîne de caractères identique à une autre. Les opérations d'éditions étaient dans un premier temps :

- l'ajout d'une lettre;
- la suppression d'une lettre;
- le changement d'une lettre (remplacement par une autre).

Dans un second article paru l'année suivante (10), Lowrance et Wagner y ajoutèrent la transposition de lettres adjacentes.

Exemple : Habittation Habitacion
en 1 suppression et 1 remplacement

Abitsaion Habitacion
en 1 ajout, 1 transposition et
1 remplacement.

Des algorithmes sont proposés dans les articles pour pratiquer les opérations nécessaires auxquelles un poids est attaché selon l'importance qu'on leur accorde.

Pour nous, l'intérêt de cette démarche est qu'on pourrait calculer une distance entre un signe et tous les mots de notre dictionnaire. Le mot proposé comme correction du signe serait celui dont la distance avec le signe est la plus courte.

Enfin, signalons que, de nous deux, c'est David Gouthière qui était le plus attiré par cette technique. Il a d'ailleurs réalisé la plus grosse partie du travail de conception et de mise en oeuvre.

III. 2. 1. 2. Calcul de la distance.

Le calcul de la distance se pratique sur deux chaînes de caractères. Dans notre cas, l'une est le signe en entrée S, c'est-à-dire un mot mal orthographié; l'autre est un mot du dictionnaire M. Ces deux chaînes différant par leurs composants et/ou leur longueur.

Exemple : Maidessinne
et
Medicina

Le calcul de la distance se résume à compter ces différences. Ces dernières sont dues à :

- des transpositions de lettres;
- des substitutions de lettres;
- des ajouts de lettres;
- des oublis ou suppressions de lettres.

Le nombre de différences constitue la distance entre les deux chaînes. Chaque opération de transformation vaut un point. Une condition toujours nécessaire pour une opération est que deux lettres de même rang soient différentes dans le signe et le mot : $S_{[i]} \neq M_{[j]}$, $i - 1$ caractères et $j - 1$ caractères ayant déjà été comparés.
Si une opération est nécessaire, il faut encore déterminer laquelle. Pour ce faire, il faût observer les caractères du mot et du signe :

- si $(S_{[i]} = M_{[j+1]}) \wedge (S_{[i+1]} = M_{[j]})$
  alors inversion
- si $(S_{[i+1]} = M_{[j+1]})$
  alors substitution
- si $(S_{[i]} = M_{[j+1]}) \wedge (S_{[i+1]} \neq M_{[j]})$
  alors oubli
- si $(S_{[i]} \neq M_{[j+1]}) \wedge (S_{[i+1]} = M_{[j]})$
  alors ajout

III. 2. 1. 3. Exemple de calcul de la distance.

Voici quelques exemples de calcul de la distance entre plusieurs signes et un mot : MEDICINA.

- MEDIICNA : une inversion
  ==> distance = 1.
- MEDISINA : une substitution
  ==> distance = 1.
- MAIDESINA : un ajout + trois substitutions
  ==> distance = 4.
- MAIDESSINA : deux ajouts + trois substitutions
  ==> distance = 5.
- MAIDESSINNE : trois ajouts + quatre substitutions
  ==> distance = 7.

III. 2. 1. 4. Algorithme.

Voici comment se présente l'algorithme de calcul de la distance appliquant les principes énoncés ci-dessus.
Le rang courant de lettre dont on s'occupe dans le mot du dictionnaire est i.

On calcule la distance entre un mot du dictionnaire et un signe entré au terminal pour chaque lettre du mot.

Si les deux lettres de rang i dans le mot et le signe sont différentes alors :

A. On vérifie si la lettre de rang i du mot est identitque à la lettre de rang (i + 1) du signe et si la lettre de rang i du signe est identique à la lettre de rang (i + 1) du mot.
   - si oui : on augmente la distance de 1.
   - si non : on continue en B.

B. On vérifie si la lettre de rang (i + 1) du mot est identique à la lettre de rang (i + 1) du signe.
   - si oui : on augmente la distance de 1.
   - si non : on continue en C.

C. On vérifie si la lettre de rang (i + 1) du signe est identique à la lettre de rang i du mot.
   - si oui : on augmente la distance de 1.
   - si non : on continue en D.

D. On vérifie si la lettre de rang i du signe est identique à la lettre de rang (i + 1) du mot.
   - si oui : on augmente la distance de 1.
   - si non : on continue en E.

E. Si on est arrivé à la dernière lettre d'un des deux mots
   - alors on augmente la distance de 1 plus le nombre de lettres supplémentaires que compte le mot le plus long par rapport au mot le plus court.
   - sinon on augmente la distance de 2.

Vous trouverez le code Pascal exécutable de cet algorithme à l'annexe **3**.

III. 2. 1. 5. Critique de cette technique.

Un reproche que l'on pourrait adresser à cet algorithme est que dans certains cas, il sélectionne comme mot le plus proche un mot qui ne ressemble en rien au signe. Par exemple, pour "assassin" ou "assassinat", il sélectionnera "casa" et pour "oreille", il sélectionnera "tortilla".

Le remède à ce problème serait de déterminer un seuil de distance au-delà duquel le programme ne proposerait pas le mot le plus proche comme orthographe correcte mais avouerait plutôt son incapacité de reconnaître le signe. Ce seuil devrait être fixé d'après l'observation du comportement du programme et de nombreux tests.

Un deuxième défaut décelé à l'expérimentation de ce programme est sa vitesse d'exécution. En effet, le calcul de la distance oblige à des manipulations relativement longues et ce, pour tous les mots du dictionnaire. Par exemple, si le temps de correction de "auga" en "agua" est négligeable (moins d'une seconde), il atteint 11 secondes pour corriger "fruita" en "fruta", 16 secondes pour changer "kilometre" en "kilometro" et plus de 30 secondes pour changer "vancanciones" en "vacaciones".
Ce temps de réponse relativement long nous a amené à abandonner ce programme comme module de correction d'orthographe, l'autre technique s'étant révélée plus rapide.

C'est dommage car la technique de la distance possède un gros avantage : quel que soit le signe, elle proposera toujours un mot qui serait sensé être sa version orthographiquement correcte. Elle ne dira jamais qu'elle ne connaît pas le mot et le corrigera toujours pour le"transformer" en un mot de son dictionnaire.
Mais cet avantage se transforme en inconvénient, comme il est dit plus haut, quand la proposition de correction est trop éloignée du signe examiné. Et le remède à cet inconvénient annule l'avantage de proposition de correction dans tous les cas.

En résumé, c'est la lenteur relative de cette technique qui nous l'a fait abandonner au profit de celle choisie. L'algorithme de calcul de la distance sera quand même employé dans une application où son temps de réponse sera réduit du fait de la faible taille de la liste de mots employée.

III. 2. 2. Méthode statistique de correction d'orthographe.

Cette technique modifie le signe traité pour tenter de le transformer en un mot appartenant au dictionnaire. Les modifications se font dans un ordre déterminé par les fréquences relatives des lettres dans le dictionnaire.

III. 2. 2. 1. Base de cette technique.

Comme son nom l'indique, l'idée de cette technique vient de l'article de Ronald W. Cornew paru en 1968, "A Statistical Method of Spelling Correction" (16). Cet article a déjà été présenté au point II. 3. 2. du présent chapitre. Rappelons qu'il s'agit d'utiliser les fréquences des digrammes dans les mots pour déterminer les lettres les plus susceptibles statistiquement d'être incorrectes dans le signe et les lettres ayant le plus de chance statistiquement de les remplacer. Cornew substitue les lettres ayant le plus de chance d'être fausses, par celles qui ont le plus de chance de les remplacer avec succès et vérifie si le mot ainsi modifié est un mot du dictionnaire. Pour plus de détails, revoyez le point II. 3. 2. ; un exemple y est détaillé.

III. 2. 2. 2. Technique employée.

Les opérations pratiquées sur le signe pour tenter de le changer en mot sont la suppression de lettre, le remplacement de lettre et l'insertion de lettre.

III. 2. 2. 2. 1. Suppression.

Tout d'abord, on essaie de trouver un mot correctement orthographié en enlevant une des lettres.

a. On enlève la lettre dont le produit des fréquences des digrammes dont elle fait partie est minimum.
b. On regarde si le mot ainsi formé existe dans le dictionnaire.
c. Si oui, on le propose comme candidat à l'orthographe correcte du signe et c'est la fin du traitement.
   Si non, on recommence en b. avec le signe original amputé de la lettre dont le produit des fréquences des digrammes est juste supérieur à celui de la lettre que l'on vient d'enlever et ce jusqu'à ce que :
   - soit on trouve un mot : on le propose comme candidat;
   - soit on arrive au bout des lettres du signe et on continue pour le remplacement.

III. 2. 2. 2. 2. Remplacement.
****

Ensuite, on essaie de transformer le signe en un mot du dictionnaire en lui remplaçant une de ses lettres.

a. On choisit la lettre dont le produit des fréquences des digrammes dont elle fait partie est minimum.
b. On choisit dans la matrice des fréquences de digrammes la lettre dont le produit des digrammes qu'elle formerait si elle était à la place choisie dans le mot est maximum.
c. On remplace dans le mot et on regarde si celui-ci ainsi modifié existe dans le dictionnaire.
d. Si oui, on le propose comme candidat à l'orthographe correcte du signe et c'est la fin du traitement.
   Si non, on recommence en c. mais en choisissant dans la matrice des fréquences des digrammes la lettre dont le produit des digrammes qu'elle formerait si elle était à la place choisie dans le mot est juste inférieur à celui de la lettre que l'on vient d'essayer et ce jusqu'à ce que :
   - soit on trouve un mot : on le propose comme candidat.
   - soit on arrive au bout de la liste des lettres de remplacement possibles : on recommence en b. en choisissant dans le signe la lettre dont le produit des fréquences de digrammes est juste supérieur à celui de la lettre que l'on vient de tenter de remplacer et ce jusqu'à ce que :
     - soit on trouve un mot : on le propose comme candidat.
     - soit on remplace toutes les lettres du mot : on continue avec l'insertion.

III. 2. 2. 2. 3. Insertion.
****

Enfin, on essaie de transformer le signe en un mot du dictionnaire en y insérant une lettre supplémentaire.

a. On choisit dans le signe le digramme dont la fréquence est minimum.
b. On y insère entre chaque composante la lettre dont le produit des fréquences des digrammes qu'elle formerait à cette place est maximum.
c. On regarde si le mot ainsi formé existe dans le dictionnaire.
d. Si oui, on le propose comme candidat à l'orthographe correcte du signe et c'est la fin du traitement.

Si non, on recommence en c. mais en choisissant pour insérer dans le signe initial la lettre dont le produit des fréquences des digrammes qu'elle formerait à cette place est juste inférieur et cela jusqu'à ce que

- soit on trouve un mot : on le propose comme candidat.
- soit on arrive au bout de la liste des lettres susceptibles d'être insérées et on recommence en b. mais en choisissant le digramme dont la fréquence est juste supérieure à celle du précédent et ce jusqu'à ce que
  - soit on trouve un mot : on le propose comme candidat.
  - soit on arrive au bout de la liste des digrammes du signe. Dans ce dernier cas et après avoir épuisé toutes les possibilités disponibles, on dit à l'utilisateur qu'on ne connaît pas ce mot et on l'invite à recommencer en écrivant un autre.

III. 2. 2. 3. Critique de cette technique.

La plus grosse lacune de cet algorithme est qu'il ne corrige qu'une seule faute dans un mot. Si par exemple il y a deux fautes d'orthographe dans un mot, une lettre manquante et une lettre erronée, cet algorithme ne pourra la corriger.
A première vue, cet inconvénient ne paraît pas trop grave car Monsieur Régimont juge peu probable que ses élèves oublient ou ajoutent plus d'une lettre ou se trompent dans plus d'une lettre dans un mot. Néanmoins, ces fautes d'orthographe peuvent également être des fautes de frappe, surtout que les élèves n'ont pas l'habitude d'écrire à l'aide d'un clavier et risquent donc d'être plus nombreuses dans un même mot. De même, certaines fautes dues à la phonétique des mots ne pourront être corrigées. Prenons l'exemple du mot "orthographe". Si l'élève écrit "orthografe", il n'y a qu'une seule faute pour l'enseignant chargé de la correction alors qu'il y en a deux pour le module de correction, deux lettres erronées ou plutôt une lettre erronée "f" et une lettre manquante "p" ou "h". La lacune est donc toujours là.

Une autre lacune existe par rapport à l'algorithme de calcul de distance : on ne se préoccupe pas ici des transpositions à l'intérieur de couple de lettres. Monsieur Régimont estime ces fautes peu probables dans le cadre des exercices écrits d'espagnol. S'il devait y en avoir, ce serait à la suite de fautes de frappe.

Or, notre expérience personnelle ainsi que celle de dactylos de notre entourage nous a appris que cette faute était relativement fréquente. Nous avons alors décidé d'ajouter une opération supplémentaire de correction qui intervertirait les lettres entre elles.

III. 2. 2. 4. Amélioration : emploi de l'interversion de lettres.

Il nous semblait que les fautes de frappe seraient plus fréquentes et que l'interversion de lettres était la plus courante parmi celles-ci. Nous avons donc décidé dans notre traitement d'un mot incorrect de tenter d'abord de le corriger en intervertissant les lettres deux à deux avant de les supprimer, remplacer ou en insérer.

Le mécanisme employé est le suivant. On intervertit d'abord entre elles la première et la deuxième lettres du signe. Si le mot ainsi formé est dans le dictionnaire,

alors on le propose comme candidat à l'orthographe correcte du signe et le traitement est terminé

sinon on remet les lettres à leur place primitive et on intervertit la deuxième et la troisième lettres ... et ainsi de suite jusqu'à ce que :

- soit on ait trouvé
- soit on ait interverti entre elles l'avant-dernière et la dernière lettres.

On remarque que dans ce cas-ci, on n'emploie pas la fréquence des digrammes pour déterminer le digramme le moins probable et donc les deux premières lettres à intervertir. La raison en est la rapidité de l'exécution. En effet, nous avons considéré, au vu de notre expérience personnelle au clavier, que l'interversion de lettres était la faute la plus fréquente et donc la plus probable. Nous plaçons donc cette opération de correction en tête afin d'accélérer la vitesse d'exécution.

La longueur moyenne des mots de notre vocabulaire étant de six caractères, nous avons préféré faire toutes les interversions (donc en moyenne 5 pour un mot de 6 lettres) de gauche à droite plutôt que de perdre du temps à consulter la matrice des fréquences des digrammes, calculer les produits des fréquences puis classer ces produits en ordre croissant.

III. 3. Localisation du traitement de modification de mot.

III. 3. 1. Fondement de la localisation du traitement.

Une de nos préoccupations majeures dans l'écriture de ce module était la vitesse d'exécution. Pour accélérer celle-ci, nous voulions profiter du passage dans le dictionnaire pour tenter de découvrir la localisation de la faute d'orthographe dans le mot, et ce en le comparant avec les membres du dictionnaire.
Le fondement de notre démarche est le suivant :

- s'il existe dans le dictionnaire au moins un mot commençant par la même première lettre que le mot traité
  - alors on continue
  - sinon il y a une faute dans la première lettre.
- s'il existe dans le dictionnaire au moins un mot commençant par les mêmes deux premières lettres que le mot traité
  - alors on continue
  - sinon il y a une faute dans les deux premières lettres.
- s'il existe dans le dictionnaire au moins un mot commençant par les mêmes trois premières lettres que le mot traité
  - alors on continue
  - sinon il y a une faute dans les trois premières lettres.

  .
  .
  .
- s'il existe dans le dictionnaire un mot ayant toutes les lettres du mot traité
  - alors on continue
  - sinon il y a une faute d'orthographe dans le mot.

Cet algorithme permet d'essayer de localiser la lettre ou la partie du mot mal orthographiée. C'est cette partie du mot que le module va modifier pour tenter de trouver un mot correctement orthographié. Elle se limite aux n premières lettres et c'est ainsi que nous la désignerons par la suite.

III. 3. 2. Traitement de faute dans les n premières lettres.

Lorsqu'on a pu déterminer qu'il y avait une faute dans les n premières lettres du mot, on applique l'algorithme suivant :

a. On recherche dans la matrice des fréquences de digrammes celles que forment les n + 1 premières lettres entre elles et on obtient n + 1 fréquences (la première lettre forme un digramme avec l'espace qui la précède).
b. On calcule le produit des digrammes pour chaque lettre, sauf la n + 1ème qui ne participe qu'au seul dernier digramme de la partie du mot qui nous intéresse, et on obtient n fréquences.
c. On essaie de retrouver un mot correct en intervertissant les (n + 1) premières lettres entre elles comme expliqué au point III. 2. 2. 4. Si on y parvient, on propose le mot ainsi construit comme correction, sinon on continue.
d. On essaie de retrouver le mot correct en enlevant une des n premières lettres, comme expliqué au point III. 2. 2. 2. 1. Si on y parvient, on propose le mot ainsi construit comme correction, sinon on continue.
e. On essaie de retrouver le mot correct en remplaçant une des n premières lettres, comme expliqué au point III. 2. 2. 2. 2. Si on y parvient, on propose le mot ainsi construit comme correction, sinon on continue.
f. On essaie de retrouver le mot correct en insérant une lettre parmi les (n + 1) premières, comme expliqué au point III. 2. 2. 2. 3. Si on y parvient, on propose le mot ainsi construit comme correction, sinon on continue.
g. Puisqu'on n'est pas parvenu à construire un mot correctement orthographié, on dit à l'utilisateur qu'on ne connaît pas le mot.

III. 4. Traitement des mots longs.

III. 4. 1. Raisons d'un traitement dédicacé aux mots longs.

Jusqu'ici, notre algorithme détecte d'abord si le mot est correctement orthographié ou pas.
Si le mot est incorrectement orthographié, on détermine le nombre maximum "n" de lettres initiales communes avec au moins un mot du dictionnaire. La faute se trouve dans les (n + 1) premières lettres du mot et plus probablement sur la (n + 1)ème.

Ensuite, on essaie de corriger la faute parmi ces (n + 1) premières lettres en les intervertissant d'abord entre elles deux à deux, puis en les supprimant l'une après l'autre, puis en les remplaçant par d'autres lettres et enfin en y insérant une lettre supplémentaire.

Les tests effectués sur cet algorithme se révélèrent concluants et satisfaisants quant à la vitesse d'exécution pour les mots de longueur courte et moyenne. Il n'en était pas de même pour les mots les plus longs, surtout si la faute se situait dans la seconde moitié du mot. Nous avons alors décidé de créer un algorithme spécial pour les mots de plus de 6 lettres. Six lettres est le seuil que nous avons estimé critique pour la longueur des mots un peu par hasard et aussi parce que le nombre de mots de plus de 6 lettres appartenant à notre vocabulaire de test n'était pas très élevé. Cette caractéristique a une grande influence sur la vitesse d'exécution. Nous avons appelé cet algorithme "escape" car à partir des lettres initiales d'un mot, il est capable de reconstruire ce mot.

III. 4. 2. Construction de l'algorithme "escape".

Cet algorithme a donc été construit pour les mots longs. Ces derniers - ont plus de 6 lettres
- peuvent contenir plusieurs fautes à condition qu'au moins les 5 premières lettres ou les 3 dernières soient justes.

Le traitement sur le signe est le suivant :

a. Si le nombre maximum de lettres communes avec au moins un mot du dictionnaire "commax" est supérieur à 4 :
   - on constitue une liste de tous les mots du dictionnaire ayant ce même préfixe de "commax" lettres.
   - on calcule la distance entre chacun d'entre eux et le signe selon la méthode décrite au point III. 2. 1.
   - on retient le mot le moins distant du signe.

b. - On détermine les trois dernières lettres du signe.
   - On constitue une liste de tous les mots du dictionnaire ayant ce même suffixe de trois lettres.
   - On calcule la distance entre chacun d'eux et le signe selon la méthode décrite au point III. 2. 1.
   - On retient le mot le moins distant du signe.

c. Si on a sélectionné un mot grâce à son préfixe, on choisit entre celui-ci et celui sélectionné grâce à son suffixe le mot le moins distant du signe. Sinon, on ne retient que ce dernier et on le propose comme correction.

III. 4. 3. Critique de cet algorithme.

Nous avons construit cet algorithme en nous appuyant sur les caractéristiques de notre vocabulaire de test. Il est bien évident qu'il peut être adapté à d'autres données ayant des caractéristiques différentes. Les premières modifications venant à l'esprit étant le nombre de lettres du préfixe (6 ici) et le nombre de lettres du suffixe (3 ici). Ces valeurs ont été choisies pour limiter au maximum la taille de la liste de mots dont on va calculer la distance d'avec le signe. Ce point est crucial en ce qui concerne la vitesse d'exécution. Ces valeurs sont donc susceptibles de changer en fonction du vocabulaire utilisé.

III. 5. Algorithme du module de correction d'orthographe.

Suite à tout ce qui précède, voici l'algorithme que nous avons retenu, David Gouthière et moi-même, pour le module de correction d'orthographe.

a. lire le signe
b. le rechercher dans le dictionnaire
c. s'il est dans le dictionnaire,
   - alors le signe est correctement orthographié et l'algorithme est terminé
   - sinon on continue en d.
d. on calcule n qui est le nombre maximum de lettres initiales communes que le signe possède avec au moins un mot du dictionnaire et on continue en e.
e. si la longueur du signe est supérieure à 6 lettres
   - alors on lui applique l'algorithme "escape" (voir III. 4. 2.)
   - sinon on continue en f.
f. on essaie de retrouver le mot correct en intervertissant entre elles les (n + 1) premières lettres (voir III. 2. 2. 4.)

si on y parvient
- alors on propose le mot ainsi construit comme correction
- sinon on continue en g.

g. on recherche la fréquence des digrammes formés par les (n + 1) premières lettres du mot entre elles.
- on calcule le produit des digrammes pour chaque lettre.
- on classe ces produits par ordre croissant.
- on essaie de retrouver le mot correct en enlevant une des n premières lettres (voir III. 2. 2. 2. 1.)

si on y parvient
- alors on propose le mot ainsi construit comme correction
- sinon on continue en h.

h. on essaie de retrouver le mot correct en remplaçant une des n premières lettres (voir III. 2. 2. 2. 2.)

si on y parvient
- alors on propose le mot ainsi construit comme correction
- sinon on continue en i.

i. on essaie de retrouver le mot correct en insérant une lettre parmi les n premières (voir III. 2. 2. 2. 3.)

si on y parvient
- alors on propose le mot ainsi construit comme correction
- sinon on continue en j.

j. on n'a pas su corriger le mot et on dit qu'on ne le connaît pas.

Vous trouverez le code Pascal exécutable de cet algorithme à l'annexe 5.

IV. Réalisation du module.

IV. 1. Introduction.

La réalisation proprement dite du module, c'est-à-dire la programmation, a été menée à bien sur les ordinateurs APPLE II de l'institut, comme il a été dit au point 2 de l'introduction. Ce faisant nous avons poursuivi trois objectifs.

Tout d'abord, nous avons construit, testé et amélioré le programme pour évaluer et augmenter la justesse des corrections qu'il apportait aux mots orthographiquement incorrects. C'est ainsi que nous avons adopté la méthode basée sur la fréquence des digrammes (voir III. 2. 2.) et que nous avons par après ajouté le traitement réservé aux mots longs (voir III. 4.).

D'un autre côté, nous avons cherché à occuper le moins de place possible en mémoire centrale afin de préserver de l'espace disponible pour le reste du didacticiel. Dans ce but, nous avons modifié la structure de données c'est-à-dire essentiellement la représentation du dictionnaire.

Enfin, nous nous sommes efforcé d'accélérer le plus possible la vitesse d'exécution du module, ce qui ne nous a pas pris le moins de temps, en utilisant notamment certaines particularités du Pascal U. C. S. D.

C'est d'ailleurs pour des raisons de lenteur d'exécution excessive que nous n'avons pas retenu la méthode de calcul de la distance pour le module. Ce sont ces deux derniers points, la diminution de la place mémoire occupée et l'augmentation de la vitesse d'exécution, que nous allons voir dans ce paragraphe à travers la structure de données et certaines particularités du code.

IV. 2. Structure de données.

Une différence sera faite entre la structure de données sur disquette et la structure de données en mémoire centrale.

IV. 2. 1. Structure de données sur disquette.

Un mot est traditionnellement représenté par une chaîne de caractères en Pascal, autrement dit un "string". Or, en Pascal U. C. S. D., le type "string" est prédéfini sur 80 caractères. Nous avons donc décidé de déclarer notre dictionnaire comme un "file of string", chaque élément de ce fichier contenant un mot. Un mot est donc représenté sur 80 caractères par un "string" en mémoire secondaire. Les éléments du dictionnaire sont classés par ordre alphabétique.

IV. 2. 2. Structure de données en mémoire centrale.

IV. 2. 2. 1. Structures de données non retenues.

Nous avons d'abord pensé représenter le dictionnaire en mémoire centrale par un tableau de "string", de la même forme que le fichier de "string" sur disquette. Ce tableau étant chargé au début de l'exécution du programme testant le module, il permettait d'effectuer une recherche dichotomique d'un mot du dictionnaire, les éléments du tableau étant classés par ordre alphabétique.

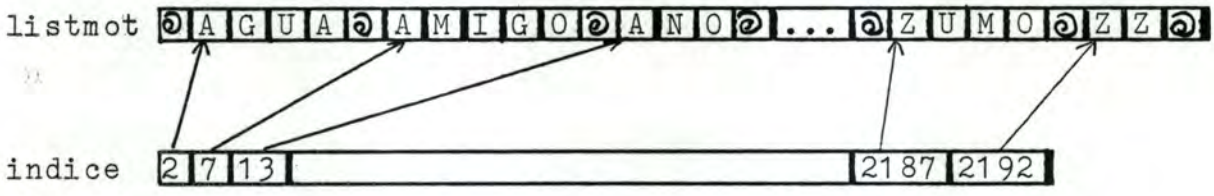
Malheureusement, cette représentation gaspille pas mal de places en mémoire centrale. En effet, il a déjà été dit qu'un "string" était défini par défaut sur 80 caractères.
Or aucun des mots du dictionnaire n'en compte tant puisqu'il y a 16 mots de 3 lettres, 44 mots de 4 lettres, 80 mots de 5 lettres, 77 mots de 6 lettres, 44 mots de 7 lettres, 29 mots de 8 lettres, 15 mots de 9 lettres, 5 mots de 10 lettres, 4 mots de 11 lettres et 1 mot de 13 lettres. Cela donne une moyenne de près de 6 lettres par mots. On perdait donc la place nécessaire pour ranger en moyenne 74 caractères par mots, soit 23.310 caractères.

Une solution était de définir un type "string" sur un nombre plus limité de caractères, ce qui est possible en Pascal U. C. S. D. Mais, pour permettre les comparaisons entre le signe en entrée et les mots du dictionnaire, il fallait que le signe soit du même type que les mots du dictionnaire. Par conséquent, la taille du "string" défini sur un nombre limité de caractères devait être assez grande pour accepter des signes trop longs. Cette taille devait également être au moins égale à 13 caractères puisqu'un mot du dictionnaire comptait 13 lettres. Mais cette option n'était pas satisfaisante car elle gaspillait encore trop de places. Songez que si on avait choisi une taille de "string" de 20 caractères, l'espace destiné à contenir en moyenne 14 caractères par mot était perdu ce qui représentait 4.270 caractères. On pouvait évidemment encore réduire la longueur des "strings" mais il y avait toujours de l'espace réservé pour rien si on gardait une longueur fixe pour tous les enregistrements de mots.

Alors, Monsieur Cherton nous a suggéré la structure de données suivante qui est celle que nous avons retenue.

IV. 2. 2. 2. Structure de données retenue.

Deux tableaux sont employés : "listmot" et "indice". "Listmot" est un "packed array of char". Il contient tous les caractères de tous les mots du dictionnaire, chaque caractère d'un article du fichier étant un élément du tableau "listmot" et chaque article du fichier étant précédé et suivi par un caractère séparateur "@" dans "listmot".
De son côté, "indice" est un tableau d'entier dont les éléments sont les indices dans "listmot" du premier caractère de chaque mot du dictionnaire. La structure de données du dictionnaire est donc représentée par le schéma suivant :



Le dernier mot de "listmot" et "indice" est "ZZ". Il est là pour permettre le chargement du dernier mot de "listmot" et "indice" dans un "string". Ce chargement nécessite de connaître la longueur du mot à transférer et cette dernière est calculée comme la différence entre l'indice du premier caractère du mot suivant et l'indice du premier caractère du mot à transférer moins un pour le caractère séparateur "@".
Par exemple, le dernier mot de "listmot" est "zumo".
Son indice est 2.187; sa longueur, 4 caractères; soit "zz" dont l'indice est 2.192 ==>longueur de "zumo" = 2192 - 2187 - 1 = 4.

IV. 2. 2. 3. Avantage de la structure retenue.

Cette représentation, occupant le moins de place mémoire possible, sera évidemment utilisée pour tout le système. Elle permet également de constituer relativement facilement la matrice des fréquences relatives des digrammes.

Au départ, nous avions constitué cette matrice manuellement. Elle se trouve d'ailleurs pour illustration à l'annexe 4. Nous ne nous étions pas préoccupés de sa constitution par le programme lorsque les mots étaient des "strings". Nous assignions simplement les valeurs que nous avions nous-mêmes calculées à chaque élément de la matrice. Pour nous, l'espace était représenté par le caractère "@" et la matrice était déclarée comme suit :
matrice : array ("@".. "Z", "@".. "Z") of integer;

Lorsque le dictionnaire a été représenté par les tableaux "lismot" et "indice", il fut très facile de calculer la matrice des fréquences des digrammes en parcourant le tableau "listmot" et en comptant le nombre de digrammes présents.
Ce calcul est effectué juste après le chargement du dictionnaire en mémoire centrale dans le tableau "listmot".

Pendant le chargement est aussi constitué et classé par ordre alphabétique le tableau des indices dans "listmot" des suffixes de trois lettres des mots longs (voir III. 4. 2.). Pour chaque mot long, on calcule l'indice de son antépénultième lettre dans "listmot" et on range cet indice à sa place appropriée dans le tableau "indiceterm". Celui-ci est classé par ordre alphabétique sur la première lettre du suffixe afin de permettre la recherche dichotomique. Il joue donc le même rôle que le tableau "indice" mais pour les suffixes de trois lettres appartenant aux mots longs.

IV. 2. 2. 4. Remarques sur l'utilisation du module.

Un dernier mot sur la place occupée en mémoire par les tableaux "indice" et "listmot". Pour éviter au maximum de perdre de la place, nous avons déclaré leur borne supérieure, respectivement "nbremot" et "nbrecar" comme égales au nombre de mots et nombre de caractères de notre vocabulaire de test soit 316 mots et 2.193 caractères (en comptant les caractères séparateurs "@" et le dernier "mot" de "listmot" : "zz").
La valeur de ces deux constantes pourra être modulée en fonction de la taille du vocabulaire employé par le programme.

"Nbremot" doit être au moins égal au nombre réel de mots du dictionnaire + 1 ("zz") et "nbrecar" doit être au moins égal au nombre réel de caractères du dictionnaire plus le nombre de séparateurs (1 par mot) + 2 ("zz") + 1 (le dernier séparateur).

Enfin, il ne faut pas oublier de modifier la procédure "chargement" si on veut utiliser le module pour traîter les mots du vocabulaire du didacticiel. En effet, tel qu'il existe actuellement, le fichier du module de correction d'orthographe est un fichier de "string". Il se trouve sur une disquette sous le nom "diction.data". Cette disquette doit se trouver dans le deuxième drive de la machine. Pour adapter le module de correction d'orthographe au dictionnaire du didacticiel, il faudra éventuellement modifier le type du fichier et les modalités d'acquisition d'un article de ce fichier en fonction de ce qui sera défini notamment pour le module de génération de phrases.

IV. 3. Particularités du code augmentant la vitesse d'exécution.

Les points cruciaux touchant à la vitesse d'exécution sont la recherche d'un mot du dictionnaire et l'accès aux différents caractères d'un mot. Le premier problème fut réglé par l'utilisation de la recherche dichotomique tandis que pour le second, on fit appel à une procédure prédéfinie en Pascal U. C. S. D.

Au départ, nous effectuions donc une recherche dichotomique sur le tableau de "string" présent en mémoire centrale, les éléments de ce tableau étant classés par ordre alphabétique. La comparaison entre le signe en entrée et un mot du dictionnaire se faisant globalement et non caractère par caractère. Le traitement était donc suffisament rapide.

Lorsqu'il dut s'exercer sur le tableau de caractères "listmot", ses performances se dégradèrent. En effet, pour accéder à un mot du dictionnaire, il fallait d'abord accéder à son indice. Ensuite, il fallait insérer ses caractères l'un après l'autre dans un "strig" intermédiaire en transformant chaque caractère en "string" de longueur 1. Toutes ces opérations successives sur les différentes lettres faisaient perdre du temps, surtout le chargement caractère par caractère et la conversion de ceux-ci en "string".

Nous avons alors tiré parti d'une procédure prédéfinie en Pascal U. C. S. D., la procédure "moveleft", qui assure le déplacement d'un nombre spécifié de bytes. Un exemple d'appel à cette procédure est

moveleft (source, destination, longueur)

où "source" et "destination" sont des variables de n'importe quel type sauf de type "file". Le premier byte de "source" est le début de l'intervalle de bytes dont les valeurs sont copiées dans l'intervalle de bytes commençant au premier byte de "destination". "Longueur" est un entier et spécifie le nombre de bytes copiés. Moveleft commence à la gauche de "source" et de gauche à droite copie les bytes dans "destination" en commençant par la gauche. Il existe aussi une procédure "moveright" travaillant de droite à gauche. Nous avons utilisé "moveleft" pour charger un mot de "listmot" dans un "string" intermédiaire. Cette opération se fait plusieurs fois dans la recherche dichotomique d'un mot du dictionnaire. Cette dernière a également lieu très souvent, chaque fois que le module cherche à savoir si le signe qu'il a construit en modifiant le signe en entrée se trouve dans le dictionnaire.
Le byte de départ dans "listmot" est donné par l'indice du premier caractère du mot se trouvant dans le tableau "indice". Puisque "listmot" est un tableau compacté, chaque caractère occupe un byte et non deux s'il n'était pas compacté. Le nombre de bytes à transférer est donné par la différence entre l'indice du mot suivant et l'indice du mot à charger, ce qui donne la longueur du mot en enlevant le caractère séparateur "ↂ". Ce nombre de bytes est copié dans le byte 0 du "string" par un "moveleft".
On s'était auparavant arrangé pour mettre dans celui-ci le caractère séparateur "ↂ" en chargeant le mot.
Cela donne : - indcar : = indice [demi]

On accède à l'indice du premier caractère du mot qui sera le byte de départ du moveleft.

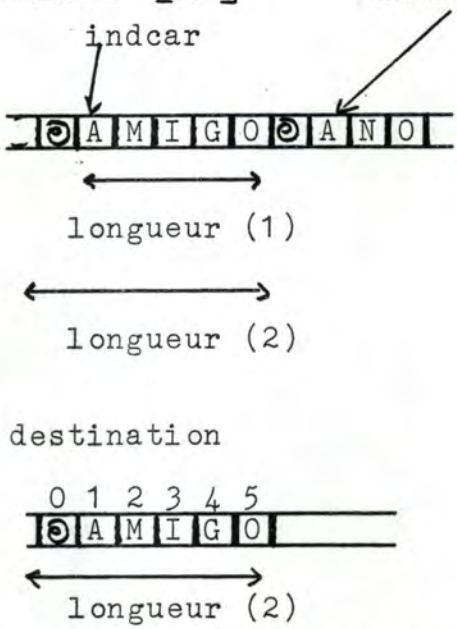
- longueur : = indice [demi + 1] - indcar - 1; (1)

On calcule le nombre de bytes à transférer en éliminant le deuxième caractère séparateur.

- longueur : = longueur + 1 (2)

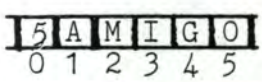
On augmente le nombre de bytes pour y incorporer le premier caractère séparateur.

- moveleft (listmot [indcar - 1] , destination, longueur)



Le mot précédé du caractère séparateur est copié dans les "longueur" premiers bytes de destination à partir du byte 0. Celui-ci est inaccessible à l'utilisateur et indique au système la longueur utilisée du "string". Comme tel, le "string" est inutilisable puisque le système pense qu'il contient une valeur sur 80 caractères.

longueur : = longueur - 1;
moveleft (longueur, destination, 1)
destination :



La longueur effective du mot est copiée dans le byte 0 du "string" et celui-ci est rendu utilisable. On trouve ici la justification de "zz" comme dernier mot du dictionnaire. L'utilisation de cette fonction a grandement amélioré la vitesse d'exécution du système de l'ordre de 2,5 à 3.

V. Traitement des mots pluriels.

V. 1. Introduction.

Les noms écrits par l'utilisateur du didacticiel peuvent être singuliers ou pluriels. Le module de correction doit donc pouvoir traiter indifféremment les mots singuliers ou pluriels. Comme indiqué au point 6 du chapitre I, la formation du pluriel des noms espagnols est relativement simple :

en règle générale, on ajoute "s" aux mots terminés par une voyelle et "es" aux mots terminés par une consonne. Profitant de cette simplicité apparente, j'ai adapté le module de correction d'orthographe pour qu'il traite les mots espagnols pluriels. Ce fut finalement plus difficile que prévu au départ.

V. 2. Principe général.

Lorsque le module reçoit le signe en entrée, il vérifie d'abord si celui-ci est au pluriel. Si oui, il le met au singulier avant de le traiter. A la fin du traitement, le mot proposé comme correction est remis au pluriel si le signe l'était à son entrée dans le module. Le traitement proprement dit ne se fait donc que sur des mots singuliers et les mots du dictionnaire ne sont qu'au singulier.

V. 3. Cas particuliers à prendre en compte.

Malheureusement, ce principe général est trop simple et passe sous silence un certain nombre de particularités de la formation du pluriel espagnol. Certains des problèmes causés par ces particularités ont pu être résolus, d'autres pas pour différentes raisons. Nous allons passer ces problèmes en revue en commençant par ceux qui n'ont pas trouvé de solution, puis les autres avec leur solution.

V. 3. 1. Cas des í accentués finaux.

La première critique concernant ce mécanisme est qu'il n'applique que la règle générale de formation du pluriel. Il ne fonctionne donc pas correctement pour les mots terminés par un "í" accentué dont le pluriel est formé en ajoutant "es".

Or, on ne pourra tenir compte des "í" accentués que lorsque le module pourra traiter les accents, les "í" accentués étant comme leur nom l'indique surmontés d'un accent aigu. A ce moment la mise au pluriel des mots terminés par un "í" accentué ne posera plus de problèmes.

V. 3. 2. Cas des autres voyelles accentuées finales.

Pour les mots terminés par une voyelle accentuée autre qu'un "í", le problème est quelque peu différent. Ils prennent seulement un "s" au pluriel, mais selon certaines sources, ils admettent les deux terminaisons.

Ne pouvant traiter les accents, le présent module traitera les voyelles accentuées comme des voyelles non accentuées et seule la formation du pluriel en ajoutant "s" sera acceptée.

V. 3. 3. Cas des noms propres terminés par "s" ou "z".

Les noms propres terminés par "s" ou "z" restent invariables au pluriel. Ce cas particulier pourra être facilement traité lorsque le module pourra distinguer les majuscules des minuscules. En attendant, il assimilera les noms propres aux noms communs. Dans le cadre de ce module, le problème n'est donc pas résolu.

V. 3. 4. Cas des mots terminés par "z".

A la formation du pluriel, le "z" final devient "c" devant "es". Par exemple, "el pez" devient au pluriel "los peces". Il faudra donc en tenir compte à la mise au singulier où, après avoir enlevé "es", on transformera le "c" final en "z" . Une critique vient tout de suite à l'esprit : que se passe-t-il à la mise au singulier de mots se terminant par "c" au singulier ? Arbitrairement, on changera leur "c" final, qui est correct, en "z" . Mais cette critique n'est pas valable dans le cadre de notre vocabulaire de test où aucun mot ne se termine par "c". A la mise au pluriel, si le mot se termine par "z", on transforme ce "z" en "c" et on ajoute "es".

V. 3. 5. Cas des polysyllabes en "s" non accentués sur la dernière.

Les polysyllabes en "s" non accentués sur la dernière restent invariables au pluriel. Malheureusement, le module ne peut reconnaître les accentuations. Néanmoins, il n'y a que 5mots se terminant par "s" dans notre vocabulaire : "apendicitis", "autobus", "gafas","mes" et "tocadiscos". Le cas de "gafas" (lunettes) est particulier. La forme singulier "gafa" existe mais n'a pas la même signification que la forme pluriel, la seule connue des élèves de Monsieur Régimont.

"Mes" est monosyllabe et forme normalement son pluriel en "meses". "Apendicitis", "autobus" et "tocadiscos" restent invariables au pluriel.

V. 3. 6. Cas des mots se terminant par "e" ou "s" au singulier.

V. 3. 6. 1. Le problème.

Lorsqu'on reçoit un signe seul en entrée, la seule solution, pour savoir s'il est au pluriel, consiste à vérifier s'il se termine par "s" ou "es" lorsqu'on ne dispose pas d'un déterminant de ce signe. La mise au singulier se réalise en enlevant "s" aux signes terminés par "s" et "es" aux signes terminés par "es". Mais si on applique ce principe, les mots se terminant par "s" ou "e" au singulier sont incorrectement traités. Par exemple, "autobus" est transformé par la mise au singulier en "autobu" qui ne se trouve pas dans le dictionnaire. Le module le corrige en lui ajoutant "s" puis la mise au pluriel considérant que le mot était au pluriel puisqu'il se terminait par "s", ajoute "es" à "autobus". Ce qui fait que recevant "autobus" correctement orthographié au singulier, le module propose comme correction une mauvaise forme pluriel "autobuses".

Le cas des mots singuliers se terminant par "e" est encore plus curieusement traité. Prenons par exemple le mot "fuente" dont le pluriel est "fuentes". A la mise au singulier, "fuentes" est transformé en "fuent" qui ne se trouve pas dans le dictionnaire et est donc considéré comme incorrectement orthographié. Le module le corrige en lui ajoutant "e" et la mise au pluriel ajoute "s" à "fuente". Donc, recevant "fuentes" correctement orthographié au pluriel, le module propose comme correction "fuentes", c'est-à-dire exactement la même chose. Cette lacune devra être supprimée.

V. 3. 6. 2. La solution.

La solution consiste à considérer la mise au singulier comme une tentative de correction du signe en entrée.

En effet, recevant le signe, le module vérifie d'abord s'il se trouve dans le dictionnaire. Si c'est le cas, le signe est correctement orthographié et l'algorithme est terminé. Sinon, le module essaie de mettre le signe au singulier. S'il y parvient, dans le cas où le signe se termine par "s" ou "es", il recherche si ce singulier est dans le dictionnaire. Si oui, le signe est correctement orthographié et sinon on essaie de le corriger par les moyens déjà vus, c'est-à-dire la procédure "escape" ou l'inversion, la suppression, le remplacement et l'insertion de lettres.

Mais la mise au singulier ne se fait pas comme expliqué ci-dessus; elle tient maintenant compte des mots se terminant par "e" au singulier. En effet, si le signe se termine par "s", on enlève celui-ci. On vérifie alors si le signe ainsi formé se trouve dans le dictionnaire. Si oui, le signe est correctement orthographié. Sinon, on enlève la dernière lettre du signe si celle-ci est un "e". Dans ce dernier cas, on recherche si le signe ainsi amputé de ses deux dernières lettres se trouve dans le dictionnaire. Si oui, le signe est correctement orthographié. Dans le cas contraire, on essaie de le corriger par les autres opérations.

Voyons comment cela se passe dans le cas de notre exemple "autobus". Recevant le signe, le module le recherche dans le dictionnaire et l'y trouve. Il signale alors que le mot est correctement orthographié. Dans le cas des mots se terminant par "s" au singulier et correctement orthographiés, le traitement est donc semblable à ce qu'il était dans le module n'intégrant pas le traitement des mots pluriels.

Prenons maintenant le cas de "fuentes". Recevant le signe, le module le recherche dans le dictionnaire et ne l'y trouve pas. Il essaie alors de le mettre au singulier. Tout d'abord, il supprime le "s" final et vérifie si le signe "fuente" se trouve dans le dictionnaire. La réponse étant positive, le module signale que le mot est correctement orthographié.

Voyons ensuite le traitement d'un mot pluriel terminé par "s" et correctement orthographié : "amigos". Recevant le signe, le module ne le trouve pas dans le dictionnaire. Il essaie alors de le corriger tout d'abord en le mettant au singulier. Il obtient par cette opération le signe "amigo" qu'il trouve dans le dictionnaire. Par conséquent, il signale que l'orthographe du mot est correcte.

Examinons enfin le cas d'un mot pluriel terminé par "es" : "vacaciones". Recevant le signe, le module le recherche dans le dictionnaire et ne l'y trouve pas. Il essaie alors de le mettre au singulier. Tout d'abord, il supprime le "s" final et vérifie si le signe "vacacione" se trouve dans le dictionnaire. La réponse étant négative, le module enlève le "e" final et vérifie si le signe "vacacion" se trouve dans le dictionnaire. La réponse étant positive, le module signale que le mot est correctement orthographié.

Le temps de traitement des mots pluriels est légèrement plus long du fait d'une ou deux recherches supplémentaires dans le dictionnaire. Toutefois, cette perte de temps est négligeable grâce à la rapidité de la recherche dichotomique.

V. 3. 7. Correction d'une mauvaise formation du pluriel.

V. 3. 7. 1. Le problème.

Qu'en est-il des mots dont le pluriel n'est pas correctement formé ? On peut ici retenir quatre fautes possibles :

- les mots pluriels terminés par une consonne et "s"
- les mots pluriels terminés par une voyelle et "es"
- les mots pluriels terminés par "zes"
- les polysyllabes pluriels terminés par "ses"

Dans chacun de ces quatre cas, un mot dont le pluriel est incorrectement formé serait considéré comme orthographiquement juste. Par exemple, "vacacions", "amigoes", "pezes" et "autobuses" seraient mis au singulier ce qui donnerait respectivement "vacacion", "amigo", "pez" et "autobus" qui sont bien l'orthographe correcte de la forme singulier. Le module dirait alors que l'orthographe du signe entré, c'est-à-dire respectivement "vacacions", "amigoes", "pezes" et "autobuses",est correct alors que cela n'est manifestement pas vrai. C'est ce problème qu'il faut résoudre.

V. 3. 7. 2. La solution.

Pour que le module prenne aussi en compte les formations incorrectes du pluriel, il faut qu'il les détecte à la mise au singulier.

Ainsi, si le signe se termine par un "s" précédé d'une consonne, la formation du pluriel est fausse.

D'autre part, si le signe se termine par "es" précédé d'une voyelle ou de "z", la formation du pluriel est également fausse.

En ce qui concerne les polysyllabes terminés par "s", c'est un peu plus compliqué. Tout d'abord, ils se terminent par "ses" quand leur pluriel est mal formé. Ensuite, comme leur nom l'indique, ils ont plusieurs syllabes. Ils comptent donc au moins quatre lettres, y compris le "s" final; la première syllabe pouvant n'être composée que d'une seule lettre, et la deuxième de trois : une consonne, une voyelle et le "s" final. Donc, si le signe est terminé par "ses" au pluriel et compte plus de quatre lettres, la formation du pluriel est fausse.

Si le signe mis au singulier n'est toujours pas correctement orthographié, le reste du module se chargera de le corriger. S'il y parvient, le mot sera remis correctement au pluriel pour être proposé comme correction.

Si le signe mis au singulier est correctement orthographié, il est remis correctement au pluriel avant d'être proposé comme correction. De cette façon, le module considère bien que les mauvaises formations du pluriel sont des fautes d'orthographe et il le manifeste en proposant la forme pluriel correcte comme solution.

Remarquons que ce procédé peut corriger deux fautes différentes et simultanées de formation du pluriel. Prenons le cas de "pez" dont le pluriel est "peces". Si le signe en entrée est "pezs", il y a deux fautes : le "s" après une consonne et "z" non transformé en "c" devant "es" au pluriel. Le module détectera la mauvaise formation du pluriel due au "s" après une consonne et la corrigera en "peces".

V. 4. Algorithme de mise au singulier.

Voici donc l'algorithme de mise au singulier d'un mot espagnol.

A. Si le signe se termine par "s"
- alors on continue en B.
- sinon le signe est au singulier et donc la formation du pluriel ne peut être incorrecte et l'algorithme est terminé.

B. Le signe est au pluriel, on enlève le "s" final et on regarde si le signe ainsi amputé se trouve dans le dictionnaire.
- si oui, le signe mis au singulier est un mot correctement orthographié et si la dernière lettre du mot singulier n'est pas une voyelle,
  - alors la formation du pluriel était incorrecte.
  - sinon la formation du pluriel était correcte et l'algorithme est terminé.
- sinon on continue en C.

C. Si le signe (amputé de son "s" final) se termine par "e", alors
- on enlève le "e" final.
- si le signe ainsi amputé du "es" final se termine par "c"
  - alors : on remplace le "c" par "z" et la formation du pluriel était correcte.
  - sinon : si le signe ainsi amputé du "es" final se termine par "z",
    - alors la formation du pluriel était incorrecte.
    - sinon : si le signe ainsi amputé du "es" final se termine par "s" et compte plus de quatre lettres,
      - alors la formation du pluriel était incorrecte.
      - sinon la formation du pluriel était correcte.
- on recherche si le signe ainsi amputé du "es" final se trouve dans le dictionnaire
  - si oui, le signe mis au singulier est un mot correctement orthographié et si la dernière lettre du mot singulier est une voyelle,
    alors la formation du pluriel était incorrecte.

V. 5. Algorithme de mise au pluriel.

Lorsque le signe a été mis au singulier, il faut le remettre au pluriel. Voici donc l'algorithme de formation du pluriel de mots espagnols.

A. Si le signe en entrée était au pluriel,
- alors, on ajoute "s" à la fin du signe et l'algorithme est terminé.
- sinon on continue en B.

B. Si la dernière lettre du signe mis au singulier est un "z",
- alors on le remplace par "c".
- sinon on continue en C.

C. Si la dernière lettre du signe mis au singulier n'est pas un "s" ou si le mot ne compte pas plus de quatre lettres, alors on ajoute "es" à la fin du signe et l'algorithme est terminé

V. 6. Algorithme du module de correction d'orthographe incorporant le traitement des mots pluriels.

Suite à tout ce qui précède, voici l'algorithme que j'ai retenu pour le module de correction d'orthographe traitant les mots espagnols pluriels.

A. lire le signe.

B. Le rechercher dans le dictionnaire.

C. s'il est dans le dictionnaire,
- alors le signe est correctement orthographié et l'algorithme est terminé.
- sinon on continue en D.

D. tenter de mettre le signe au singulier si la forme singulier est dans le dictionnaire,
- alors si la formation du pluriel était correcte,
    - alors le signe est correctement orthographié et l'algorithme est terminé.

- sinon on remet le signe au pluriel, on propose cette forme comme correction et l'algorithme est terminé.

- sinon on continue en E.

E. On calcule n qui est le nombre maximum de lettres initiales communes que le signe possède avec au moins un mot du dictionnaire et on continue en F.

F. si la longueur du signe est supérieure à 6 lettres
- alors on lui applique l'algorithme "escape" (voir III. 4.2.) et on passe en K.
- sinon on continue en G.

G. on essaie de retrouver le mot correct en intervertissant entre elles les (n + 1) premières lettres (voir III.2. 2. 4.).
Si on y parvient
- alors on passe en K.
- sinon on continue en H.

H. - on recherche la fréquence des digrammes formés par les (n + 1) premières lettres du mot entre elles.
- on calcule le produit des digrammes pour chaque lettre.
- on classe ces produits par ordre croissant.
- on essaie de retrouver le mot correct en enlevant une des n premières lettres (voir III. 2. 2. 2. 1.)
- si on y parvient :
  - alors on passe en K.
  - sinon on continue en I.

I. on essaie de retrouver le mot correct en remplaçant une des n premières lettres (voir III.2. 2. 2. 2.)
Si on y parvient :
- alors on passe en K.
- sinon on continue en J.

J. on essaie de retrouver le mot correct en insérant une lettre parmi les n premières (voir III.2. 2. 2. 3.)
Si on y parvient :
- alors on passe en K.
- sinon on continue en L.

K. si le signe en entrée était au pluriel, on met la proposition de correction au pluriel, on la soumet à l'utilisateur et l'algorithme est terminé.

L. on n'a pas su corriger le mot et on dit qu'on ne le connaît pas.

Vous trouverez les spécifications et le code Pascal exécutable de cet algorithme aux annexes 7 et 8.

BIBLIOGRAPHIE DU CHAPITRE V.

(1)James L. PETERSON, Computer Program for Spelling Correction: An Experiment in Program Design, Berlin Heidelberg, Springer Verlag, 1980. (p. 2)

(2)L. DAVIDSON, Retrieval of Misspelled Names in an Airlines Passenger Record System, Communications of A. C. M. Volume 5, Numéro 3, Mars 1962. (p.169 - 171)

(3)G.CARLSON, Techniques for Replacing Characters that are Garbled on Input, Proceedings of the I966 Spring Joint Computer Conference, 1966. (p.183 - 193)

(4)D. N. FREEMAN, Error Correction in CORC : The Cornell Computing Language, PH. D. Thesis, Department of Computer Science, Cornell University, Septembre 1963.

(5)C. K. Mc ELWAIN and M. E. EVANS, The Degarbler - A Program for Correcting Machine - Read Morse Code, Information and Control, Volume 5, Numéro 4, Décembre I962. (p.368 -384)

(6)F.J. DAMERAU, A Technique for Computer Detection and Correction of Spelling Errors, Communications of the A. C. M. Volume 7, Numéro 3, Mars 1964. (p. 171 - 176)

(7)C. N. ALBERGA, String Similarity and Misspellings, Communications of the A. C. M., Volume 10, Numéro 5, Mai 1967. (p. 302 - 313)

(8)E. M. RISEMAN et R. W. EHRICH, Contextual Word Recognition Using Binary Digrams, IEEE Transactions on Computers, Volume C 20, Numéro 4, Avril 1971. (p. 397 - 403)

(9)R. A. WAGNER et M. J. FISCHER, The String-to-String Correction Problem, Journal of the A. C. M., Volume 21, Numéro 1, Janvier 1974. (p. 168 - 173)

(10)E. M. RISEMAN et A. R. HANSON, A Contextual Postprocessing System for Error Correction Using Binary n-Grams, IEEE Transactions on Computers, Volume C 23, Numéro 5, Mai I974. (p. 480 - 493)

(11)R. LOWRANCE et R. A. WAGNER, An Extension to the String-to-String Correction Problem, Journal of the A. C. M., volume 22, numéro 2, Avril 1975.(p. 175 - 183)

(12)R. MORRIS et L. L. CHERRY, Computer Detection of Typographical Errors, IEEE Transactions on Professionnal Communications, Volume PC 18, numéro 1, Mars 1975. (p. 54 - 64)

(13)L. E. MAC MAHON, L. L. CHERRY et R. MORRIS, Statistical Text Processing, The Bell System Technical Journal, Volume 57, Numéro 6, 2° partie, Juillet - Août 1978. (p. 2137 - 2154)

(14)L. D. HARMON, Automatic Reading of Cursive Script, Proceedings of a Symposium on Optical Character Recognition, Spartan Books, Janvier 1962. (p. 151 - 152)

(15)C. M. VOSSLER et N. M. BRANSTON, The Use of Context for Correcting Garbled English Text, Proceedings of the 19th A. C. M. National Convention, Août 1964. (p. D2.4-1 à D 2.4-13)

(16)R. W. CORNEW, A Statistical Method of Spelling Correction, Information and Control, Volume 12, Numéro 2, Février 1968. (p. 79 - 93)

(17)H. KUCERA et W. N. FRANCIS, Computational Analysis of Present-Day American English, Brown University Press, 1967.

(18)F. J. DAMERAU, A Technique for Computer Detection and Correction of Spelling Errors, Communications of the A. C. M. Volume 7, Numéro 3, Mars 1964. (p. 171 - 176)

(19)J.-L. DOLBY, An Algorithm for Variable-Length Proper-Name Compression, Journal of Library Automation, Volume 3/4, Décembre 1970. (p. 257 - 275)

(20)James L. PETERSON, Computer Programs for Detecting and Correcting Spelling Errors, Communications of the A. C. M., Volume 23,. Numéro 12, Décembre 1980. (p. 676 - 687)

****

C H A P I T R E V I.

LE MODULE DE GENERATION DE PHRASES.

Introduction.

Comme il est expliqué au chapitre II, tous les exercices du didacticiel ont ceci en commun qu'ils proposent à l'élève soit de compléter des phrases dans les exercices de type 1, 2 et 3, soit de former des phrases dans les exercices de type 4 et 5. Nous avons vu au chapitre IV que toutes ces phrases peuvent être décomposées comme suit :

phrase : : = <nom> <verbe conjugué> <préposition> <nom>

La conjugaison étant assurée par un autre module que la génération de phrases, la structure des phrases se ramène ici à :

phrase : : = <nom> <verbe> <préposition> <nom>

Les phrases espagnoles obéissant à cette structure constituent donc des questions potentielles du didacticiel.
Le second paragraphe du chapitre IV a expliqué les avantages d'un module de génération automatique de phrases reposant sur un modèle de sémantique. On a également vu pourquoi ce modèle de sémantique devait être aussi utilisé pour un module de correction de la sémantique des phrases.
Mais ce point ne nous intéresse pas dans le présent chapitre qui va traiter exclusivement du module de génération automatique de phrases, module dont la faisabilité est capitale pour l'automatisation des scéances d'exercices telles que la verrait Monsieur Régimont.

I. Le problème.

Le problème posé par la génération de phrases sémantiquement correctes a déjà été abordé au chapitre IV.
Rappelons-en les points les plus importants.

Deux options sont possibles : le stockage de toutes les phrases utilisées par le didacticiel dans un fichier et la génération de phrases par programme.

Pour des raisons exposées au second paragraphe du chapitre IV, la seconde option a été choisie. Cette option suppose la construction d'un modèle de sémantique des phrases du didacticiel. Ce modèle sera employé à la fois par la génération de phrases et le contrôle sémantique de phrases. Le module de génération sera employé pour générer les phrases servant de support aux exercices de type 1, 2, 3 et 4. Pour éviter à l'enseignant le fastidieux travail d'invention de phrases, le module devra pouvoir générer aléatoirement des phrases selon la structure définie : <sujet> <verbe> <préposition> <lieu>.
La préposition sera toujours "A" ou "EN". Ces phrases doivent être grammaticalement correctes et sémantiquement acceptables. Elles ne doivent employer que les verbes et les noms appartenant au vocabulaire défini par Monsieur Régimont comme celui que sont sensés connaître ses élèves. La liste des verbes se trouve au paragraphe 5 du chapitre I. La liste de tous les substantifs se trouve à l'annexe 2. Tous les déterminants de noms sont admis, aussi bien les articles définis et indéfinis que les démonstratifs et les possessifs, dans les limites des règles grammaticales réglementant leur emploi. La liste de ces déterminants se trouve à l'annexe 9. La conjugaison des verbes ne nous occupe pas ici et fera l'objet d'un module séparé. Il suffit simplement de savoir que plusieurs temps sont possibles dans le cadre du didacticiel, comme déjà dit au paragraphe 5 du chapitre I : l'indicatif présent, l'indicatif imparfait, l'indicatif futur, l'indicatif passé, le subjonctif présent et l'impératif.

II. Idées de solution.

II. 1. Introduction.

L'idée de départ était de constituer d'abord des classes de noms et des classes de verbes. Ensuite, on aurait déterminé quelles combinaisons entre classes étaient toujours possibles et lesquelles étaient toujours impossibles.

II. 2. Traits de sous-catégorisation de noms.

Avant d'entamer ce travail, j'ai consulté plusieurs ouvrages traitant de grammaires génératives, notamment "Principes de grammaire générative" de Jos Nivette (1), "Description générative et transformationnelle de la langue française" de Jean Le Galliot (2) et "Initiation méthodique à la grammaire générative" (3) et "Grammaire générative : hypothèses et augmentations" (4) de Christian Nique.
Ces lectures m'ont amené à essayer de classifier les substantifs du vocabulaire proposé selon les catégories proposées par Jean Le Galliot pour la sous-catégorisation des noms (5) :
± humain, ± animé, ± commun, ± propre, ± abstrait, ± concret, ± collectif, ± individuel, ± nombrable, ± masculin et ± mâle.
D'autre part, les catégories des verbes sont ± transitifs, ± attributifs, ± avoir (auxiliaire), ± actif, ± duratif, ± perfectif, ± sujet humain et ± objet animé, catégories auxquelles j'ai ajouté bien évidemment deux autres :"demande la préposition "EN""et"demande la préposition "A"".
J'ai ensuite essayé de faire rentrer tous les mots du vocabulaire proposé par Monsieur Régimont. Mais les difficultés rencontrées pour attribuer certaines catégories ainsi que le peu de clarté et de précision dans la définition des titres de celles-ci m'ont vite convaincu que cette démarche ne mènerait pas bien loin. De plus ces catégories ne sont pas adaptées aux cas particuliers de la srtucture de phrase <sujet> <verbe> <préposition> <complément de lieu> .
Un professeur de français que j'ai consulté n'est pas arrivé à un résultat satisfaisant. Vous pouvez d'ailleurs en juger vous-même en consultant l'annexe 8. De même, Monsieur Régimont n'est pas parvenu à faire mieux. J'ai donc abandonné cette démarche.

II. 3. Détermination des phrases à partir des verbes.

Je me suis alors dit que je pourrais travailler plus directement sur le vocabulaire en essayant de combiner des mots avec un verbe. J'ai procédé de la manière suivante : pour chaque verbe, j'ai fait la liste de tous les substantifs qui pourraient en être le sujet. Ces listes se trouvent à l'annexe 9.

Ensuite, pour chaque verbe, j'ai fait la liste de tous les noms qui pouvaient être le complément de lieu. Ces listes se trouvent à l'annexe 12.
Mon idée était de déterminer l'ensemble des compléments de lieu pouvant être attribués à un sujet et un verbe.
Le module aurait d'abord généré un sujet, puis en fonction de celui-ci un verbe, puis enfin en fonction du sujet et du verbe un complément de lieu. Cette méthode avait le grand avantage de permettre de générer un très grand nombre de phrases sémantiquement correctes. Malheureusement, elle me prit également beaucoup de temps pour constituer les listes de sujets et de compléments de lieu.
En effet, "vivir" admettait 92 sujets et 86 compléments de lieu,
"trabajar", 84 sujets et 96 compléments de lieu,
"meterse", 96 sujets et 88 compléments de lieu,
"comer", 90 sujets et 86 compléments de lieu,
"morir", 92 sujets et 88 compléments de lieu,
"pasar", 99 sujets et 89 compléments de lieu,
"marcharse", 92 sujets et 158 compléments de lieu,
"quedar", 139 sujets et 93 compléments de lieu,
"ir", 104 sujets et 158 compléments de lieu,
"llegar", 105 sujets et 116 compléments de lieu,
"subir", 108 sujets et 81 compléments de lieu,
"nadar", 80 sujets et 53 compléments de lieu,
"venir", 108 sujets et 81 compléments de lieu,
"estar", 263 sujets et 131 compléments de lieu,
"situarse", 257 sujets et 108 compléments de lieu.
Pour chaque verbe, il me fallait donc examiner si chaque sujet admettait chaque complément de lieu. Par exemple, dans le cas de "estar", il me fallait examiner 263 X 131 = 21.253 phrases possibles.

Devant l'ampleur du travail qui m'attendait, j'ai cherché un autre méthode. Celle-ci pourrait donner des résultats mais le temps m'était compté. Quelqu'un de plus familiarisé que moi dans la phraséologie, surtout espagnole, pourrait peut-être mener ce travail à bien.

De cette manière, il serait certain que le système puisse générer toutes les phrases possibles conformes à la structure exigée et employant le vocabulaire fixé.

Malgré l'abandon de cette méthode, elle me fut utile dans la mesure où les résultats qu'elle avait fournis me servirent dans la suite.

II. 4. Constitution de classes homogènes.

Finalement, j'ai appliqué la première idée qui était l'établissement de classes de mots homogènes. Mais, contrairement à ma première démarche où les classes étaient définies en dehors du contexte du vocabulaire fixé, j'ai ici constitué mes classes à partir de l'observation que j'ai faite de celui-ci.
Voici chronologiquement comment j'ai procédé.

II. 4. 1. Constitution de classes sujets.

Comme j'avais commencé par le verbe "vivir" (vivre) pour faire la liste de ses sujets possibles, j'ai repris dans cette liste les sujets humains et les animaux parce que le deuxième verbe avec lequel j'avais travaillé était "trabajar" (travailler) et que tous les animaux ne travaillent pas. A ces deux classes, il fallait ajouter les végétaux qui sont également des êtres vivants. J'obtenais ainsi trois classes qui reprenaient les sujets de "vivir" (vivre) et "trabajar" (travailler) mais aussi ceux de "comer" (manger), "morir" (mourir), "marcharse" (partir) et "nadar" (nager) : humains, animaux et végétaux.
Mais dans ces trois classes n'étaient pas repris d'autres mots représentant des sujets animés non vivants et sujets de "meterse" (s'introduire), "pasar" (passer), "ir" (aller), "llegar" (arriver), "subir" (monter) et "venir" (venir) : "agua" (eau), "autobus" (autobus), "coche" (voiture), "máquina" (machine), "mar" (mer), "río" (rivière, fleuve), "sol" (soleil) et "tren" (train), c'est-à-dire des sujets animés non vivants.
Enfin il restait les sujets inanimés de "estar", les plus nombreux. Mais cette catégorie rassemblait des mots tels que "almacén" (entrepôt), "aparador" (buffet), "armario" (armoire), "avenida" (avenue), "balcón" (balcon), "barrio" (quartier), "brasero" (foyer), "brocha" (brosse), "cafetera" (cafetière), "caja" (caisse), "calle" (rue), "cama" (lit), "camisa" (chemise), "carbon" (charbon), "carcel" (prison), "carne" (viande), "carta" (lettre), "cartel" (affiche), "casa" (maison), "cerveza"

(bière), "churro" (beignet), "cine" (cinéma), "cinturon" (ceinture), "cita" (rendez-vous), "ciudad" (cité), "cobre" (cuivre)...
Cette classe n'était pas assez homogène et demandait à être redécomposée. En effet, comment trouver un complément de lieu qui soit valable à la fois pour "barrio" (quartier) et "camisa" (chemise), pour "cita" (rendez-vous) et "brocha" (brosse)...
De plus,ces sujets n'étaient valables que pour les verbes de situation "estar", "situarse" et éventuellement "quedarse".
Me disant que cela me prendrait trop de temps de la décomposer, J'ai abandonné cette classe de sujets inanimés disparate, me réservant le droit d'y revenir peut-être plus tard, si les autres ne me suffisaient pas pour générer suffisamment de phrases différentes ou si j'en avais le temps.

II. 4. 2. Constitution de relations entre classes sujets et verbes.

Ensuite, j'ai déterminé pour chaque verbe les catégories de sujets qu'ils admettaient. Ici, le verbe "situarse" m'a occasionné quelques problèmes et j'ai dû faire appel à Monsieur Régimont pour m'aider. A la suite de ce travail, j'ai décidé de laisser tomber la classe "sujets végétaux" qui ne comprenait que deux mots : "arbol" (arbre) et "flor" (fleur).
De plus, seul le verbe "estar" l'admettait comme sujet.
En effet, une phrase comme "l'arbre vit à la campagne" me semblait peu satisfaisante au niveau du sens.

II. 4. 3. Constitution de classes lieux.

Je me suis alors attelé à la construction de catégories homogènes pour les compléments de lieu. Comme je l'avais fait pour la constitution des classes sujets, je suis parti de la liste des compléments de lieu de "vivir" et j'ai déterminé ainsi des classes lieux : lieux d'habitation, lieux de géographie humaine, lieux de travail (ou lieux publics), meubles, lieux naturels et noms propres de lieux. Je pouvais à ce stade fusionner les deux classes "lieux de géographie humaine" et "noms propres de lieux" en une seule, mais je m'y suis refusé pour garantir l'homogénéité des classes et donc la construction éventuelle de nouvelles relations entre classes sujets, verbes et classes de lieux.

L'intitulé de la classe "lieux de travail" demande quelques explications : en fait, cette catégorie a été plus spécifiquement construite à partir des compléments de lieu de "trabajar" (travailler). Ce n'est qu'après que j'ai estimé que l'intitulé de la classe ne reflétait pas assez bien son contenu pour la construction de phrases sémantiquement correctes et que la classe est devenue "lieux publics".

Etant donné les classes sujets retenues à ce stade, j'estimais que ces classes de lieu suffisaient. Néanmoins, Monsieur Régimont, que j'ai consulté, a lui-même construit de nouvelles classes, à la fois sujets et lieux. J'en parlerai plus loin pour respecter la chronologie de ce travail.

II. 5. Construction de phrases.

Ayant auparavant déterminé quelle(s) catégorie(s) de sujet admettai(en)t chaque verbe, il ne me restait plus qu'à sélectionner les classes de lieux convenant pour chaque couple "classe sujet-verbe". Ce travail ne s'est pas fait lui non plus tout seul et m'a même posé de nombreux dilemmes quant à la signification correcte ou incorrecte de certaines combinaisons "classe sujet-verbe - classe lieu". Cela m'a également permis de rendre les classes un peu plus homogènes notamment en changeant des mots de classe ("carcel" (prison) de lieu d'habitation est devenu "lieu public") ou en en supprimant d'autres ("esquina" (coin de rue, d'une maison) dans géographie humaine ou "arbol" (arbre) dans lieu naturel).
D'autre part, j'ai été obligé de construire deux classes de lieux pour le verbe "nadar" (nager) et ce pour des raisons de facilité : une s'utilisant avec "nadar" suivi de la préposition "en" et une s'utilisant avec "nadar" suivi de la préposition "a". Il me fut également très difficile de trouver des classes de lieu pour les sujets animaux. Pour cette raison, j'ai créé une classe "sujets aquatiques" qui pourrait plus spécifiquement convenir pour "nadar". Après examen, il m'a semblé que les animaux ne pouvaient former des phrases qu'avec "meterse" (s'introduire) et des lieux d'habitation. En effet, normalement, les animaux ne vivent pas dans des lieux d'habitation. Un baudet de vit pas dans une avenue, une souris dans une cité, un quartier ou une terrasse, une vache dans un buffet et un taureau dans une source.

Il en était de même avec les autres verbes, ce qui montre la difficulté qu'occasionne la construction de classes homogènes, une souris étant finalement très différente d'un taureau ou d'un baudet. Néanmoins, on verra plus loin que j'étais trop restrictif et que Monsieur Régimont pensait qu'on pouvait être plus laxiste dans l'acceptation des phrases. Certaines paraissent peut-être tirées par les cheveux mais sont pour lui tout à fait acceptables.
La catégorie des sujets animés non vivants m'a également occasionné des difficultés, l'eau étant très différente d'un autobus ou du soleil. Cela m'a amené à construire deux nouvelles classes : les moyens de transport ("autobus", "coche" et "tren") et les éléments naturels animés ("agua", "mar", "rio" et "sol") en abandonnant le mot "maquina" (machine) qui ne rentrait dans aucune de ces deux classes.
Finalement, la difficulté rencontrée pour trouver des compléments de lieu convenables et vraisemblables pour la classe "éléments naturels animés" m'a incité à abandonner celle-ci en tant que classe sujet.

Une grosse difficulté dans la construction de ce modèle est de décider si une phrase, générée avec les représentants de classes de mots, est convenable ou pas. En effet, la structure de phrase "sujet - verbe - préposition - lieu" ne se rencontre que peu souvent dans la langue parlée ou écrite. Il y a souvent un ou plusieurs mots supplémentaires qui explicitent l'idée ou précisent un élément. Néanmoins cette structure de phrase se rencontre quand même, surtout pour des exemples de grammaire ou de vocabulaire à l'usage des écoliers et des étudiants.
Mais l'absence de contexte pour les phrases ne facilitait pas les choses, bien au contraire. Ainsi, si la phrase "la vache nage vers la branche" peut paraître bizarre elle se justifie tout à fait dans le contexte d'un récit relatant une inondation et ses conséquences. Il serait peut-être d'ailleurs souhaitable d'éliminer les phrases dont la compréhension dépend d'un contexte absent ici et de ne garder que les phrases se suffisant à elles-mêmes pour être comprises convenablement et ne pas prêter à critique sur leur contenu.

Le didacticiel étant principalement destiné à des séances de rattrapage, l'élève aura déjà suffisamment de difficultés.

Enfin, mon ignorance en matière d'espagnol était un obstacle supplémentaire de belle taille. En effet, j'étais obligé de travailler sur les traductions françaises pour juger de la cohérence des phrases que je construisais, et comme en bien des cas traduction égale trahison, certaines phrases qui me semblaient incorrectes en français ne l'étaient peut-être pas en espagnol et vice versa.

Cette crainte s'est révélée fondée quand j'en ai parlé avec Monsieur Régimont, du moins sur l'incorrection de phrases françaises par rapport aux espagnoles.
Monsieur Régimont a estimé que j'étais beaucoup trop restrictif et qu'il ne fallait pas avoir peur de générer des phrases un peu bizarres. Les élèves leur attribueront de toute façon un contexte. De plus, il m'a aidé à constituer de nouvelles classes : les aliments et les matières premières comme classes sujets et les vêtements, les récipients et l'eau à la fois comme classes sujets et comme classes lieux. Il m'a également aidé à déterminer de nouvelles phrases en étant beaucoup moins restrictif que moi. Mais il a beaucoup plus l'habitude que moi de ce genre de travail et ses conceptions et ses directives furent précieuses dans l'élaboration et des classes et des phrases.

III. La solution : les phrases retenues.

Voyons maintenant le résultat de ce travail d'élaboration du modèle. Je donnerai d'abord les classes sujets avec leurs composants.
Ensuite, pour chaque verbe, on verra les classes sujets qu'ils acceptent. Puis viendront les classes lieux.
Et nous finirons par la combinaison de phrases correctes par sujets et par chaque classe sujet par verbe.

III. 1. Sujets.

1. Sujets humains.

amigo - artista - asesino - ciego - comisario - dentista - doctor - dueño - empleado - esposa - esposo - familia - gente - gitano - hombre - huésped - jefe - mecánico - médico - mudo - mujer - negro - nieto - niño - novio/a-padre - pintor - policía - portera - primo - señor - sereno - sobrina - sobrino - tío/a - torero - vecino - loco - viejo - tonto/a -

Pronoms sujets : yo - tu - él, ella, usted - nosotros/as - vosotros/as - ellos, ellas, ustedes.

Noms et prénoms : Carmen - Elena - Pedro - Juan - Juana - Manuel - Jaime - Iago - Pepe - Pepa - Jose - Juanito - Juanita - Pepito - Pepita - el Senor López - la Senora López - Beatriz - Lolita - Fermín - Paquito - Paquita - Doña Joaquina.

2. Sujets animaux.

burro - gato - pájaro - perro - ratón - toro - vaca.

3. Moyens de transport.

autobús - metro - coche - tren - barco -

4. Sujets aquatiques.

pescado - pez -

5. Aliments.

agua - cafe - carne - cerveza - churro - cocido - col - fresa - fresón - fruta - garbanzo - huevo - jamón - judías - manzana - naranja - paella - pan - patata - tapa - tomate - tortilla - verdura - vino -

6. Vêtements.

camisa - cinturón - falda - mantilla - sombrero - traje - zapato -

7. Récipients.

caja - jarro - maceta - recipiente - vaso -

8. Eau.

agua

9. Matières premières.

carbón - cobre - cuero - lena -

- Sujets de vivir : sujets animaux , humains et aquatiques.
- Sujets de trabajar : sujets humains.
- Sujets de meterse : sujets humains et animaux.
- Sujets de comer : sujets humains, animaux et aquatiques.
- Sujets de morir : sujets humains, animaux et aquatiques.
- Sujets de pasar : sujets humains, animaux, moyens de transport et aquatiques.
- Sujets de quedar : sujets humains, animaux, aquatiques, moyens de transport.
- Sujets de estar : tous.
- Sujets de marcharse : sujets humains, animaux, aquatiques et moyens de transport.
- Sujets de ir : sujets humains, animaux et aquatiques.
- Sujets de llegar : sujets humains, animaux, aquatiques, moyens de transport.
- Sujets de subir : sujets humains, animaux, moyens de transport.
- Sujets de venir : sujets humains, animaux, aquatiques, moyens de transport.
- Sujets de nadar : sujets humains, animaux, aquatiques.
- Sujets de situarse : tous.
- Sujets de pasar un mes, una semana, un rato : aliments, vêtements, récipients, eau, matières premières.
- Sujets de quedar : aliments, vêtements, récipients, eau, matières premières.

III. 3. Classes de lieux.

1. Lieux d'habitation.

casa - cocina - comedor - cuarto - desván - edificio - habitación - hospital - mesón - palacio - patio - piso - portal - recibidor - sala - sótano -

2. Lieux de géographie humaine (agglomération).

avenida - barrio - ciudad - parque - provincia - pueblo - región -

3. Lieux publics, lieux de travail.

almacén - autobús - cárcel - cine - estación - garage - iglesia - mercado - molino - plaza - puente - puerto - restaurante - taller - tasca - teatro - terraza - tren - café -

4. Immeubles.

aparador - armario - cama - cómoda - mesa - silla - brasero -

5. Lieux naturels.

campo - costa - fuente - mar - montaña - monte - playa - río - sierra -

6. Lieux de nadar (en).

agua - mar - río -

7. Lieux de nadar (a).

amigo - arbol - ciudad - costa - fuente - gente - habitacion - pared - playa - puente - puerto -

8. Noms propres.

España - Bélgica - Italia - Inglaterra - Portugal - Suiza - Alemania - Madrid - Valencia - Zarragoza - Valladolid - Barcelona - Bruselas - Roma - París - la Sierra Nevada - los Alpes - los Pireneos - Marruecos - Africa -

9. Vêtements.

voir sujets.

10. Récipients. : Voir sujets.

11. Eau.

Voir sujets.

III. 4. Combinaison de phrases correctes.

III. 4. 1. Sujets humains.

- vivir : - lieux d'habitation
  - lieux de géographie humaine
  - noms propres de lieux
  - vêtements
- trabajar : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - noms propres de lieux
  - eau
- meterse : - lieux d'habitation
  - lieux publics
  - noms propres de lieux
  - eau
- comer : - lieux d'habitation
  - lieux publics
  - récipients
- morir : - lieux d'habitation
  - lieux de travail (publics)
  - noms propres de lieux
- pasar : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux
  - eau

  (pasar un rato, un mes, las vacaciones)
- quedar : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux
  - vêtements
  - eau

- estar : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux
  - vêtements
  - eau
- marcharse : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux
  - vêtements
  - récipients
  - eau
- ir : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - meubles
  - lieux naturels
  - noms propres
  - vêtements
  - récipients
  - eau
- llegar : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres
  - eau
- subir : - lieux d'habitation
  - lieux de géographie humaine
- venir : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres
  - eau
  - vêtements
  - récipients

- nadar : "en" : agua - mar - rio -
  "a" : amigo - arbol - ciudad - costa - fuente - gente - habitation - pared - playa - puente - puerto -
- situarse : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - noms propres
  - eau

III. 4. 2. Sujets animaux.

- vivir : - lieux d'habitation
  - lieux de géographie humaine
  - noms propres
- meterse : - lieux d'habitation
  - lieux naturels
  - lieux publics
  - eau
- comer : - lieu d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - récipients
- morir : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
- pasar un mes / un rato / una semana : - lieux d'habitation
  - lieux de géographie humaine
  - eau
- quedar : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - noms propres
  - lieux naturels
  - eau
- estar : idem

- marcharse : - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres
  - eau
  - récipients
- ir : idem
- llegar : idem
- subir : - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres
- venir : idem que pour ir, llegar et marcharse
- nadar : "en" : agua - mar - rio -
  "a" : amigo - arbol - ciudad - costa - fuente - pared - playa - puente - puerto -
- situarse : idem que pour estar et quedar

III. 4. 3. Moyens de transport.

- quedar : - lieux de géographie humaine
  - noms propres de lieux
  - lieux naturels
- pasar : - lieux de géographie humaine
  - noms propres de lieux
  - lieux naturels
- marcharse : - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux
- estar : - lieux de géographie humaine
  - lieux naturels
  - noms propres de lieux
- ir : - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux

- venir : - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux
- llegar : - lieux de géographie humaine
  - lieux publics
  - lieux naturels
  - noms propres de lieux

III. 4. 4. Pez.

- vivir : agua - mar - rio - fuente -
- comer : agua - mar - rio -
- morir : agua - mar - rio -
- pasar : agua - mar - rio -
- quedar : agua - mar - rio -
- estar : agua - mar - rio -
- marcharse : costa - fuente - playa - puerto -
- ir : arbol - costa - fuente - playa - puerto - barco -
- llegar : arbol - brazo - costa - fuente - playa - puerto -
- venir : costa - fuente - playa - puerto - barco - puente - ciudad - pueblo -
- nadar : "en" : agua - mar - rio - fuente -
  "a" : arbol - brazo - costa - fuente - gente - habitation - pared - playa - puente - puerto -

III. 4. 5. Sujets aliments.

- pasar un mes, una semana, un rato :- lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - meubles
  - lieux naturels
  - noms propres de lieux
  - récipients
- quedar : idem
- estar : idem
- situarse : idem

III. 4. 6. Sujets_vêtements.

- pasar un mes, una semana, un rato :
  - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - meubles
  - lieux naturels
  - noms propres de lieux
- quedar : idem
- estar : idem
- sutuarse : idem

III. 4. 7. Sujets_récipients.

- pasar un mes, una semana, un rato :
  - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - meubles
  - lieux naturels
  - noms propres de lieux
- quedar : idem
- estar : idem
- situarse : idem

III. 4. 8. Agua_/_eau.

- pasar un mes, una semana, un rato :
  - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics
  - meubles
  - lieux naturels
  - récipients
- quedar : idem
- estar : idem
- situarse : idem

III. 4. 9. Matières_premières.

- pasar un mes, una semana, un rato :
  - eau
  - récipients
  - lieux d'habitation
  - lieux de géographie humaine
  - lieux publics

- meubles
- lieux naturels
- noms propres de lieux

- quedar : idem
- estar : idem
- situarse : idem

IV. Le dictionnaire et son contenu.

IV. 1. Introduction.

La phrase à générer par le module est composée de mots. Ces mots seront contenus dans un dictionnaire, selon la structure de données définie au point IV. 2. 2. du chapitre V. Mais certaines caractéristiques des mots de ce dictionnaire doivent être mémorisées pour la génération de phrases. Voyons ce que sont ces caractéristiques.

IV. 2. Les mots du dictionnaire.

La structure de la phrase est <sujet> <verbe conjugué> <préposition> <lieu>. Les verbes seront conjugués par un module approprié. Donc, seuls les verbes à l'infinitif seront dans le dictionnaire. En ce qui concerne la préposition, elle ne peut être que "A" ou "EN". Elle ne sera donc pas mémorisée au dictionnaire. Les mots du dictionnaire peuvent donc être sujet, verbe ou lieu.

IV. 3. Caractéristiques des verbes.

Examinons d'abord le cas du verbe.
On a défini une phrase correcte comme une relation classe sujet-verbe-classe lieu. Les verbes sont donc considérés individuellement. Toutefois, il faut savoir quelle préposition générer. Et comme la préposition dépend du verbe, ceux-ci seront également divisés en classe selon la préposition qu'ils demandent :

préposition EN : estar, pasar, vivir, situar(se), trabajar, meter(se), comer, quedar, quedar(se), morir.
préposition A : ir, llegar, marchar(se), subir, venir.
préposition A ou EN : nadar.

Il existe donc trois classes de verbes. Seule cette caractéristique sera nécessaire pour la génération de phrases. Une restriction toutefois : je ne me suis pas préoccupé de la conjugaison et si certaines caractéristiques sont indispensables pour conjuguer correctement les verbes, il faudra les ajouter. J'espère qu'il n'en sera rien.

IV. 4. Caractéristiques des sujets et des lieux.

Voyons maintenant le cas du sujet et du lieu. En fait, ce sont les mêmes caractéristiques, avec pour seule différence la fonction : sujet ou lieu.

IV. 4. 1. Un sujet (ou lieu) peut être nom ou pronom = un nom peut être commun ou propre.

La différence entre un nom et un pronom, c'est que ce dernier n'admet pas de déterminant. Il en est de même entre un nom commun et un nom propre, par exemple un prénom ou le nom d'une ville, puisque notre matériel ne dispose pas encore des majuscules. La première caractéristique d'un sujet et d'un lieu est donc la possibilité ou non d'un déterminant, pour pouvoir le générer éventuellement.

IV. 4. 2. Un sujet (ou lieu) peut être masculin ou féminin.

Cette caractéristique est valable pour les noms communs dans le cas qui nous occupe. Elle servira à accorder le déterminant en genre avec son déterminé. La deuxième caractéristique est donc le genre du mot.

IV. 4. 3. Un sujet (ou lieu) peut avoir un pluriel ou pas de pluriel.

Certains mots, les noms propres, n'admettent pas de pluriel. De même, la forme des pronoms au pluriel est particulière. Cette caractéristique indique la possibilité de mettre le mot au pluriel si la génération a choisi de mettre le sujet ou le lieu au pluriel. Cette opération sera faite selon les règles exposées au paragraphe VI du chapitre I et au paragraphe V du chapitre V grâce à la procédure déjà définie "miseplur".

IV. 4. 4. Un sujet (ou lieu) appartient à une ou plusieurs classes.

Un mot peut appartenir à plusieurs classes différentes, ce qui augmente ses chances d'être généré. Les différentes classes auxquelles appartient un mot constituent donc sa dernière

caractéristique. On aurait pu définir des classes étant à la fois sujet et lieu mais cela compliquait inutilement la structure de données. Retenons donc qu'un mot peut appartenir à plusieurs classes différentes.

IV. 5. Caractéristiques des mots du dictionnaire.

Un mot du dictionnaire a donc cinq caractéristiques qu'il faudra retenir pour la génération de phrases :

- son orthographe : l'écriture du mot en toutes lettres.
- sa possibilité d'avoir un déterminant.
- son genre.
- sa possibilité d'être mis au pluriel.
- sa ou ses classes.

V. Représentation du dictionnaire en mémoire centrale.

Pour des raisons de vitesse d'exécution, le dictionnaire devra être chargé en mémoire centrale. Voici la description de la structure de données employée.

V. 1. L'orthographe des mots.

Pour la représentation des mots proprement dits, j'ai évidemment repris la structure de données utilisée pour la correction d'orthographe. Cette structure est décrite en détail au point IV. 2. 2. du chapitre V. Rappelons-en les points principaux.

Les mots du dictionnaire sont stockés dans le tableau de caractères "listmot". Ils y sont classés par ordre alphabétique. Chaque mot est entouré par deux caractères séparateurs "ↂ". Chaque caractère d'un mot du dictionnaire est un élément du tableau "listmot". Le dernier mot de "listmot" est "ZZ". L'indice du premier caractère de chaque mot de "listmot" est pointé par les éléments d'un tableau de record appelé "indice". Contrairement à la structure de données de la correction d'orthographe, les indices des mots ne sont pas stockés dans un seul tableau d'entiers mais dans trois tableaux de record que l'on va décrire. A part cela, la structure de données est similaire.

V. 2. Les caractéristiques des mots.

Un mot peut être sujet, verbe ou lieu. Les caractéristiques des mots sont donc contenues dans trois listes, selon la fonction du mot dans la phrase : la liste des sujets, la liste des verbes et la liste des lieux. La liste des sujets et la liste des lieux ne différant que par leur appellation, nous les décrirons ensemble. La liste des verbes sera décrite ensuite.

V. 2. 1. Liste des sujets et des lieux.

Les caractéristiques des sujets sont contenues dans le tableau "listsujet". Celles des lieux sont dans le tableau "listlieu". La structure de ces tableaux est identique :

- indice : indice dans "listmot" du premier caractère du mot concerné.
- genre : booléen - vrai si le mot est masculin, faux si le mot est féminin.
- possibilité de pluriel : booléen - vrai si le mot peut être mis au pluriel, faux sinon.
- possibilité de déterminant : booléen - vrai si le mot doit être accompagné d'un déterminant, faux sinon.
- classe : numéro de la classe du mot.

"Listsujet" et "listlieu" sont donc des tableaux de record à cinq composantes. Plusieurs éléments de "listsujet" ou de "listlieu" peuvent avoir le même indice. Cela signifie qu'un même mot peut appartenir à plusieurs classes différentes, sujet ou lieu.
Pour générer un sujet ou un lieu, un nombre aléatoire sera généré. Ce nombre sera l'indice d'un élément de "listsujet" ou "listlieu" et cet élément sera le sujet ou le lieu.

V. 2. 2. Liste des verbes.

Nous avons vu au point IV. 3. du présent chapitre que la seule caractéristique des verbes à prendre en compte était sa classe, c'est-à-dire la préposition qu'il demande.

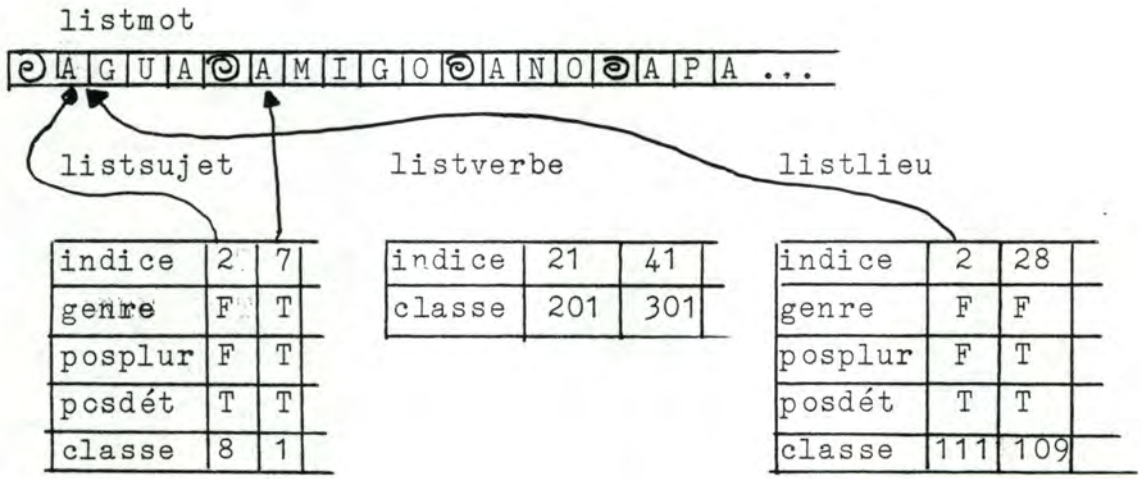
La structure du tableau "listverbe", tableau de record reprenant les caractéristiques des verbes, est donc :

- indice : indice dans "listmot" du premier caractère du verbe concerné.
- classe : numéro de la classe du verbe.

Pour générer un verbe, on procédera de la même façon que pour le sujet et le lieu. On générera un nombre aléatoire qui sera l'indice d'un élément de "listverbe" et cet élément sera le verbe.

V. 3. Schéma de la structure du dictionnaire.

Voici donc le schéma de la structure des données représentant le dictionnaire :



VI. Représentation du dictionnaire en mémoire secondaire.

Le dictionnaire devra être sauvé sur disquette lorsque le didacticiel n'est pas en fonction. La structure de données choisie pour ce faire sera un peu différente de celle choisie pour la mémoire centrale, car elle ne doit pas se plier aux mêmes contraintes. En voici la description.

VI. 1. Le fichier dictionnaire.

Tout le dictionnaire est contenu dans le fichier "dictionnaire". Celui-ci est déclaré comme un fichier de "typmot". "Typmot" est un type de donnée record défini comme ayant les parties suivantes :

- mot : chaîne (c'est-à-dire "string" de 20 caractères maximum).
- genre : booléen, vrai si le mot est masculin, faux sinon.
- posplur : booléen, vrai si le mot peut être mis au pluriel, faux sinon.
- posdét : booléen, vrai si le mot doit avoir un déterminant, faux sinon.
- classe : numéro d'une classe du mot.
- ptrclasse : pointeur vers un élément du tableau des classes supplémentaires.

VI. 2. Le fichier des classes supplémentaires.

J'ai dit que tout le dictionnaire était contenu dans le fichier "dictionnaire". Ce n'est pas tout à fait vrai. En effet, le dictionnaire contient une seule fois chaque mot et ses caractéristiques y compris sa classe. S'il appartient à plusieurs classes, celles-ci sont regroupées dans un tableau des classes supplémentaires. La partie "ptrclasse" du "typmot" est un pointeur vers un élément de ce tableau qui contient le numéro de la deuxième classe du mot. Si "ptrclasse" est nul, le mot n'appartient qu'à une seule classe.

Décrivons maintenant le tableau des classes supplémentaires. Il est chargé en mémoire centrale avant le chargement du fichier "dictionnaire" et est sauvé sur disquette après le sauvetage de "listmot", "listsujet", "listverbe" et "listlieu" sur disquette. En mémoire secondaire, il est représenté par un fichier de "classup" et en mémoire centrale par un tableau de "classup". "Classup" est un type de donnée record défini comme ayant les parties suivantes :

- numéro : numéro de classe.
- clasptr : pointeur vers un autre élément du tableau des classes supplémentaires.

Donc, si un mot appartient à trois classes différentes, le numéro de la première de ces classes sera dans la partie "classe" de "typmot", la partie "ptrclasse" de "typmot" contiendra l'indice dans le tableau des classes supplémentaires d'un élément contenant dans sa partie "numéro" le numéro de la deuxième classe du mot et dans sa partie "clasptr" un autre pointeur. Ce dernier sera l'indice d'un élément du tableau des classes supplémentaires contenant dans sa partie "numéro" le numéro de la troisième classe du mot. Comme c'est la dernière classe du mot, la partie "clasptr" du même élément est mise à zéro.

Voici un schéma représentant un exemple : le mot "agua".

typmot

mot	agua
genre	false
posplur	false
posdét	true
classe	8
ptrclasse	1

tableau des classes supplémentaires.

numéro	clasptr
10	2
111	0

VII. Algorithme de la génération de phrases.

VII. 1. Algorithme général.

L'algorithme général du module de génération de phrase est le suivant :

- générer aléatoirement un sujet.
- générer aléatoirement un verbe convenant pour la classe du sujet.
- générer aléatoirement un complément de lieu convenant pour le couple sujet-verbe.

Il s'agit donc en fait de trois appels à trois sous-modules différents. Nous allons examiner ceux-ci plus en détails.

VII. 2. Génération du sujet.

Un sujet peut être un nom commun, un nom propre ou un pronom. S'il est un pronom, il n'est pas exprimé. S'il est un nom propre, il n'a pas de déterminant et ne peut être mis au pluriel. S'il est un nom commun, il devra généralement être accompagné d'un déterminant et pourra éventuellement être mis au pluriel. On aura besoin de sa classe pour pouvoir générer le reste de la phrase. Sachant tout cela, les grands pas de l'algorithme de génération du sujet seront :

- choisir aléatoirement un sujet.
- choisir aléatoirement son nombre si c'est possible. Si c'est le pluriel, mettre le sujet au pluriel.
- choisir aléatoirement le déterminant du sujet, s'il en faut un, accordé en genre et en nombre avec ce dernier.
- donner la classe du sujet.

VII. 3. Génération du verbe.

Le verbe devra être choisi en fonction de la classe du sujet. Il devra également être conjugué à la personne demandée par celui-ci, au singulier ou au pluriel. On aura besoin de sa classe pour savoir quelle préposition générer. Enfin, son identification servira, couplée avec la classe du sujet, à générer le lieu.

Sachant cela, voici les grands pas de l'algorithme de génération du verbe :

- choisir aléatoirement un verbe en fonction de la classe du sujet.
- déterminer la préposition convenant au verbe et la générer.
- donner l'identification du verbe.

La conjugaison du verbe se fait en-dehors de cet algorithme.

VII. 4. Génération du lieu.

Le lieu devra être choisi en fonction du couple classe "sujet-verbe". Un lieu peut être un nom commun, un nom propre ou un pronom. S'il est un pronom ou un nom propre, il n'a pas de déterminant et ne peut être mis au pluriel. S'il est un nom commun, il devra généralement être accompagné d'un déterminant et pourra éventuellement être mis au pluriel.

Sachant tout cela, les grands pas de l'algorithme de génération du lieu seront :

- choisir aléatoirement un lieu accepté par le couple classe sujet-verbe.
- choisir aléatoirement son nombre si c'est possible. Si c'est le pluriel, mettre le lieu au pluriel.
- choisir aléatoirement le déterminant du lieu, s'il en faut un, accordé en genre et en nombre avec ce dernier.

VII. 5. Génération du déterminant.

Cette procédure n'est pas appelée dans l'algorithme principal mais dans la génération du sujet et dans celle du lieu. Elle consiste à choisir aléatoirement entre un article défini, un article indéfini, un démonstratif ou un possessif. Il faut ensuite générer le déterminant qui s'accorde en genre et en nombre avec son déterminé. Remarquons que pour les démonstratifs et les possessifs, il faut encore choisir entre respectivement trois et quatre possibilités. Il y a en effet trois nuances dans les démonstratifs : proche, éloigné et très éloigné. De même, il y a cinq formes possibles de possessifs selon la personne du possesseur et non six car la forme de la troisième personne est identique pour le singulier et le pluriel.

VIII. Représentation du modèle de sémantique des phrases en mémoire centrale.

Le modèle de sémantique des phrases ayant été défini, il fallait encore le mettre sous forme informatique.
Il s'agit donc de représenter les classes sujets, les verbes et les classes lieux ainsi que les relations entre ces différents éléments en mémoire.

VIII. 1. Représentation des éléments du modèle : classes et verbes.

Nous avons déjà vu au paragraphe V du présent chapitre comment étaient représentés les mots du dictionnaire.

Leurs caractéristiques, notamment le numéro de la (ou les) classe(s) à laquelle(auxquelles) ils appartiennent, sont rassemblées dans trois listes : "listsujet", "listverbe" et "listlieu". Accédant à un mot, on connaît donc immédiatement la classe à laquelle il appartient pour les sujets et lieux ou son identifiant, c'est-à-dire son indice dans "listverbe", pour les verbes. Il n'y a donc aucune difficulté de ce côté et il n'y a rien à ajouter à ce qui existe déjà.

VIII. 2. Représentation des relations entre classes.

Il faut également représenter les relations entre les différentes classes et verbes. Il s'agit de retenir qu'une relation "classe sujet - verbe - classe lieu" fournit une phrase correcte ou pas. La relation est donc booléenne.

Cela m'a donné l'idée de construire une matrice booléenne en trois dimensions. La première dimension serait la classe sujet, la deuxième le numéro du verbe et la troisième la classe lieu. Un élément de cette matrice serait vrai si la relation existant entre ses indices fournit une phrase sémantiquement correcte et faux si la phrase n'était pas sémantiquement correcte.

Cette matrice est déclarée comme suit :

```
const bornatrice = 16 { bornes de la matrice }

type correct = packed array
               [1 .. bornatrice, 1 .. bornatrice, 0 .. bornatrice]
                                          of boolean;

var correctsem : correct;
```

La matrice "correctsem" admettra donc un nombre de classes sujet compris entre 1 et 16, un nombre de verbes compris entre 1 et 16 et un nombre de classes lieux compris entre 0 et 16. Cette borne inférieure nulle alors que les autres sont égales à un demande quelques explications.
Lorsqu'on générera un sujet, on connaîtra son numéro de classe. En fonction de ce dernier, on générera alors un verbe qui soit acceptable pour cette classe sujet.

Il faut donc une indication disant qu'il existe au moins une phrase "classe sujet - verbe - classe lieu" ayant cette classe sujet et ce verbe, que la classe sujet peut être sujet de ce verbe. L'élément d'indice [classe sujet, verbe, 0] sera cette indication. Ainsi, ayant le numéro de classe sujet et un numéro de verbe, on vérifiera immédiatement que ce début de phrase est sémantiquement correct et qu'on peut générer le lieu. Si on n'avait pas cette indication, on n'aurait pas pu vérifier si la phrase était sémantiquement correcte avant de l'avoir générée tout entière.

IX. Représentation du modèle de la sémantique des phrases en mémoire secondaire.

Le sauvetage sur disquette du modèle de la sémantique des phrases nécessite une structure de données différente de celle définie en mémoire centrale. C'est cette structure de données que nous allons examiner dans le présent paragraphe.

Nous avons déjà parlé de la représentation sur disquettes des classes de mots et des verbes au paragraphe VI. Nous n'y reviendrons pas ici.

En ce qui concerne les relations entre classes, il faut trouver une représentation simple et peu coûteuse en termes de place. La première idée pourrait être de sauver la matrice booléenne "correctsem" sur un fichier de booléen. Il s'agit effectivement d'une représentation simple et peu coûteuse en place. Elle présente néanmoins un inconvénient capital.

Les trois dimensions de la matrice "correctsem" sont définies de 1 au nombre de classes sujets
1 au nombre de verbes
1 au nombre de classes lieux.

Admettons que "correctsem" soit sauvé sur un fichier de booléen. Ces éléments sont chargés un par un dans le fichier. Puis, grâce au système destiné à cette opération, le professeur ajoute deux classes sujets, un verbe et trois classes lieux. Au chargement de "correctsem", les dimensions de celles-ci ont changé depuis son sauvetage et les valeurs sauvées ne sont plus chargées à leur place initiale. De plus, certains éléments de "correctsem" auront une valeur indéterminée.

Voyons ce raisonnement sur un exemple simple.

Soit "correctsem" : tableau dont les indices sont définis de 1 à 2, 1 à 2, 1 à 2 et les éléments ont les valeurs suivantes :

correctsem	[1, 1, 1]	: T
"	[1, 1, 2]	: F
"	[1, 2, 1]	: T
"	[1, 2, 2]	: T
"	[2, 1, 1]	: F
"	[2, 1, 2]	: T
"	[2, 2, 1]	: F
"	[2, 2, 2]	: F

Ces valeurs sont sauvées dans cet ordre dans le fichier de booléen.
Puis, on ajoute une classe lieu.
On a alors "correctsem" : tableau dont les indices sont définis : de 1 à 2, de 1 à 2, de 1 à 3.
Au chargement, les éléments de "correctsem" prendront les valeurs suivantes :

correctsem	[1, 1, 1]	: T
"	[1, 1, 2]	: F
"	[1, 1, 3]	: T
"	[1, 2, 1]	: T
"	[1, 2, 2]	: F
"	[1, 2, 3]	: T
"	[2, 1, 1]	: F
"	[2, 1, 2]	: F
"	[2, 1, 3]	: indéterminé
"	[2, 2, 1]	: indéterminé
"	[2, 2, 2]	: indéterminé
"	[2, 2, 3]	: indéterminé

On remarque que certains éléments ont changé de valeur alors que l'on n'a pas modifié la sémantique des phrases : correctsem [1, 2, 2] , correctsem [2, 1, 2] , correctsem [2, 2, 1] , et correctsem [2, 2, 2] . De même, certains éléments, les nouveaux dont le dernier indice est 3, ont vu leur valeur arbitrairement fixée alors que, la sémantique de phrases n'ayant pas été modifiée, leur valeur est F.

Pour remédier à ce problème, on sauvera "correctsem" sur le fichier "phrase correcte" déclaré comme suit :

```
type seman = packed record
        valeur : boolean; (vrai si la phrase est sémantiquement
                  correcte, faux sinon)
        sujet : integer; (numéro de la classe sujet du modèle
                de phrase)
        verbe : integer; (identifiant du verbe du modèle de
                phrase)
        lieu : integer; ( identifiant du lieu du modèle de
      end;      phrase)
var phrase correcte : file of seman
```

Au chargement d'un article du fichier "phrase correcte" dans le tableau "correctsem", on met la valeur dans l'élément d'indice sujet, verbe, lieu :

```
    with phrase correcte ↑ do
        correctsem [sujet, verbe, lieu] : = valeur
```

Au sauvetage d'un élément du tableau "correctsem" dans le fichier "phrase correcte", on met la valeur de l'élément dans "valeur" et les trois indices respectivement dans "sujet", "verbe" et "lieu" :

```
    with phrase correcte ↑ do
       begin
          valeur:= correctsem [inds, indv, indl] ;
          sujet : = inds;
          verbe : = indv;
          lieu : = indl;
       end;
```

BIBLIOGRAPHIE DU CHAPITRE VI.

(1) Jos NIVETTE, Principes de grammaire générative, Bruxelles, Labor, et Paris, Nathan, 1974.

(2) Jean LE GALLIOT, Description générative et transformationnelle de la langue française, Paris, Nathan, 1975.

(3) Christian NIQUE, Initiation méthodique à la grammaire générative, Paris, Librairie Armand Colin, 1974.

(4) Christian NIQUE, Grammaire générative : hypothèses et argumentations, Paris, Librairie Armand Colin, 1974.

C H A P I T R E V I I .

LE SYSTEME DE GESTION DU DICTIONNAIRE ET DU MODELE DE SEMANTIQUE DES PHRASES.

Introduction.

Le module de génération de phrases repose sur le dictionnaire qui reprend toutes les caractéristiques de chaque mot nécessaires pour le système et sur la matrice booléenne des phrases indiquant si un modèle de phrase est ou non sémantiquement correct. Un modèle de phrase se présente comme la relation "classe sujet - verbe - classe lieu".

Pour des raisons de généralisation et d'extensibilité du système, il serait souhaitable que ces deux structures de données soient définissables et modifiables par un utilisateur non informaticien. Dans la plupart des cas, cet utilisateur sera le professeur responsable des exercices de rattrapage.
Dans ce but, j'ai construit un système permettant de gérer le dictionnaire des mots et le modèle de sémantique.
Ce système permet de

- créer des classes de mots
- obtenir la liste des classes de mots
- ajouter des mots au dictionnaire
- supprimer des mots au dictionnaire
- modifier des mots du dictionnaire
- consulter des mots du dictionnaire
- obtenir la liste des mots du dictionnaire
- ajouter des modèles de phrases au modèle de sémantique
- retirer des modèles de phrases au modèle de sémantique
- obtenir une liste des modèles de phrases sémantiquement correctes.

Ce système est finalement plus difficile et plus long à réaliser que le système de génération de phrases lui-même.

I. Gestion du dictionnaire.

I. 1. Introduction.

La gestion du dictionnaire recouvre en fait la gestion de deux fichiers : le dictionnaire proprement dit tel qu'il est décrit aux paragraphes V et VI du chapitre VI et le fichier des classes de mots existantes. Ce dernier n'existe en fait que pour pouvoir gérer le fichier dictionnaire.
Nous commencerons par le décrire lui et les opérations qu'on peut lui appliquer avant de parler de la gestion du dictionnaire proprement dit.

I. 2. Le fichier des classes de mots.

I. 2. 1. Utilité d'un tel fichier.

Nous avons décrit au paragraphe IV du chapitre VI le contenu du dictionnaire, c'est-à-dire les mots et leurs caractéristiques. Parmi celles-ci, la seule qui soit commune à la fois aux noms et aux verbes est le numéro de la classe à laquelle le mot appartient. Il faut donc d'abord définir ces classes. Telle est l'utilité d'un fichier des classes de mots.

I. 2. 2. Description d'une classe.

Dans un mot, une classe est représentée par son numéro. C'est donc la première caractéristique à retenir. C'est également par son numéro que la phrase est représentée dans un modèle de phrase. Malheureusement, le numéro d'une classe ne donne à l'utilisateur aucune information sur son contenu.
Nous avons vu au point II. 4. du chapitre VI que les classes étaient représentées pour leur concepteur par leur titre. Ce dernier doit être suffisamment parlant pour recouvrir tous les mots de la classe.
La seconde caractéristique d'une classe à retenir pour la gestion du dictionnaire est donc son titre. Pour les verbes, il existe trois classes : préposition "en", préposition "a" et indifférent.

Pour ces raisons, j'ai donc déclaré un type

```
classe = record;
   numéro : integer;
   titre : chaîne  (* string limité à 20 caractères *)
end;
```

I. 2. 3. Représentation du fichier en mémoire centrale.

En mémoire centrale, la liste des classes de mots existantes est représentée par un tableau déclaré comme suit :

```
const  dimclasex  = 30;
var clasexist : packed array (0 .. dimclasex ) of classe;
```

Ce tableau est chargé en mémoire centrale quand on en a besoin durant l'exécution du système de gestion de fichiers. Il est sauvé par après sur un fichier se trouvant sur disquette. Les éléments sont classés par ordre de numéro de classe croissant.

I. 2. 4. Représentation du fichier en mémoire secondaire.

En mémoire secondaire, la liste des classes de mots existantes est représentée par un fichier déclaré comme suit :

```
var classemot : file of classe;
```

Ce fichier est chargé en mémoire centrale dans le tableau quand on en a besoin durant l'exécution du système de gestion des fichiers. Le tableau est sauvé par après quand on n'en a plus besoin dans le fichier. Les éléments sont classés par ordre de numéros de classes croissants.

I. 3. Gestion du fichier des classes de mots.

I. 3. 1. Introduction.

Plusieurs opérations peuvent être définies sur le fichier des classes de mots. Pour ma part, je n'en ai réalisé que deux par manque de temps et parce que c'étaient les seules dont j'avais un besoin absolu : ajouter des classes de mots et obtenir un listing du fichier.

D'autres opérations pourraient être définies comme la suppression ou la modification de classes de mots. Mais ces opérations impliquaient des changements importants de toute la base de données. Imaginez que vous supprimez la classe numéro 2 "animaux". Tous les mots appartenant à cette classe devront avoir une de leurs caractéristiques, le numéro de classe, modifiée ou être carrément supprimée. La situation est analogue avec la modification. Si vous modifiez le numéro ou le titre d'une classe, les mots appartenant à cette classe devront changer de classe. Et que penser d'une phrase définie comme ayant une classe sujet qui n'existe plus ou qui est modifiée. Cette modification peut rendre la phrase tout à fait farfelue ou à tout le moins sémantiquement incorrecte.

Pour toutes ces raisons, je n'ai réalisé que la création (ou l'ajout) de classes de mots et la consultation de la liste des classes de mots.

I. 3. 2. Remarque sur la numérotation des classes.

Afin de différencier les classes sujets des classes lieux et des classes verbes, la numérotation des classes s'est faite comme suit :

- classes sujets : numéros compris entre 1 et 99
- classes lieux : numéros compris entre 101 et 199
- classe "verbe demandant la préposition "EN"" : numéro 201
- classe "verbe demandant la préposition "A"" : numéro 301
- classe "verbe demandant les 2 prépositions" : numéro 401

Les classes verbes sont prédéfinies dans le système et ne sont pas créées par la création de classes de mots.

I. 3. 3. Ajout de classes de mots.

L'utilisation de cette procédure se déroule comme suit. A chaque création de classe, le système demande

"Désirez-vous créer : 1. une classe sujet
2. une classe lieu
3. aucune classe"

Si l'utilisateur répond "1", le système lui affichera la liste des classes sujets déjà créées. S'il n'y en a pas encore, cette liste sera bien évidemment vide. Le système attribuera d'office un numéro à la classe à créer. Ce numéro sera celui de la dernière classe sujet précédemment créée, augmenté de un. L'utilisateur n'a plus alors qu'à donner son titre. Cela fait, la question demandant ce que l'utilisateur désire créer réapparaît.
Si l'utilisateur répond "2", le système lui affichera la liste des classes lieux déjà créées. S'il n'y en a pas encore, cette liste sera bien évidemment vide. Le système attribuera d'office un numéro à la classe à créer. Ce numéro sera celui de la dernière classe lieu précédemment créée, augmenté de un. Cela fait, la question demandant ce que l'utilisateur désire créer réapparaît.
Si l'utilisateur répond "3", la procédure se termine.

Vous trouverez les spécifications et le code Pascal de cette procédure appelée "creclasse" en annexe.

I. 3. 4. Consultation de la liste des classes de mots.

L'utilisateur peut demander la liste des classes des mots à l'écran ou à l'imprimante. Cette liste sera classée par ordre de numéro de classe croissant. Elle se terminera donc toujours par les classes de verbes. Chaque élément de la liste comprend le numéro de la classe et son intitulé.

La procédure visualisant la liste des classes à l'écran s'appelle "classexist". Celle imprimant cette liste s'appelle "impresclas". Vous en trouverez les spécifications et le code Pascal en annexe.

I. 4. Le fichier dictionnaire.

I. 4. 1. Introduction.

Le dictionnaire dont on s'occupe ici est celui défini pour la génération de phrases. Son contenu est décrit au paragraphe V du chapitre VI. Sa représentation en mémoire secondaire est bien évidemment identique pour la génération de phrases et la gestion de fichier. Elle est décrite au paragraphe VI du chapitre VI. La représentation en mémoire centrale est par contre légèrement différente pour faciliter la gestion des différentes listes.

I. 4. 2. Représentation en mémoire centrale.

I. 4. 2. 1. L'orthographe des mots.

La représentation des mots proprement dits est une combinaison des structures de données définies pour la correction d'orthographe et la génération de phrases.
En voici la description.

Les mots du dictionnaire sont stockés dans le tableau de caractères "listmot". Ils y sont classés par ordre alphabétique. Chaque mot est entouré par deux caractères séparateurs "@". Chaque caractère d'un mot du dictionnaire est un élément du tableau "listmot". Le dernier mot de "listmot" est "ZZ".

L'indice du premier caractère de chaque mot est un élément appelé "indlis" d'un record. Tous les records sont classés par ordre de "indlis" croissant dans un tableau.
Cette représentation est similaire à celle définie pour la correction d'orthographe avec deux tableaux : "listmot" identique à celui-ci et "indice", tableau d'entiers contenant les indices des premiers caractères de chaque mot. Ici, "indice" est un tableau de records dont une des parties est un entier contenant l'indice du premier caractère d'un mot.
"Indice [23] . indlis " contient donc l'indice dans "listmot" du premier caractère du 23ème mot du dictionnaire.

Mais la représentation définie pour la génération de phrases est également retenue ici. L'indice du premier caractère de chaque mot de "listmot" est pointé par les éléments d'un tableau de record appelés "indice". Les indices des mots sont stockés dans trois tableaux de record "listsujet", "listverbe" et "listlieu".

I. 4. 2. 2. Les caractéristiques des mots.

La représentation des caractéristiques des mots est identique à celle définie pour la génération de phrases et décrite au point V. 2. du chapitre VI. Nous n'y reviendrons donc pas.

Il y a quand même une légère différence. Nous avons déjà décrit la partie "indlis" des records contenus dans le tableau "indice". L'autre partie de ces records s'appelle "ptrtyp". C'est un caractère qui peut avoir quatre valeurs :

- "S" : le mot concerné appartient à une classe sujet et ses caractéristiques sont dans "listsujet".
- "V" : le mot concerné est un verbe et ses caractéristiques sont dans "listverbe".
- "L" : le mot concerné appartient à une classe lieu et ses caractéristiques sont dans "listlieu".
- "D" : le mot concerné appartient à une classe sujet et une classe lieu et ses caractéristiques sont dans "listsujet" et "listlieu" à la fois.

Si "indice [23].ptrtyp = "s", cela signifie que le 23ème mot du dictionnaire appartient à une classe sujet.

J'ai défini cette structure de données pour faciliter la consultation du dictionnaire. En accédant à chaque élément de "indice", on peut accéder à chaque mot grâce à "indlis" et on sait où se trouve ses caractéristiques grâce à "ptrtyp".

I. 4. 2. 3. Schéma de la structure du dictionnaire.

Voici donc le schéma de la structure de données représentant le dictionnaire :

listmot

| | | |

I. 5. Gestion du dictionnaire.

I. 5. 1. Introduction.

Plusieurs opérations peuvent être menées sur le dictionnaire :
- l'ajout de mots
- la suppression de mots
- la modification de mots
- la consultation de mots
- la consultation des listes de mots

Elles permettent à l'enseignant de créer et modifier à sa guise le dictionnaire employé par le didacticiel. En voici une description.

I. 5. 2. Ajout de mots.

Le système demande à l'utilisateur d'écrire en toutes lettres le mot à ajouter. Si le mot existe déjà au dictionnaire, il est refusé; sinon, il est accepté. Le système demande alors son genre : "m" pour masculin, "f" pour féminin.
Ensuite, l'utilisateur tape "O" si le mot peut être mis au pluriel et "N" dans le cas contraire. Après, on demande si le mot peut avoir un déterminant : "O" pour oui et "N" pour non.
Enfin, l'utilisateur écrit le numéro de classe et le système demande à chaque fois s'il y en a encore une. Rappelons qu'un mot peut appartenir à plusieurs classes.
Sur la moitié inférieure de l'écran est affichée la liste des classes de mots existantes pour permettre à l'utilisateur de choisir la classe de mots appropriée. Entre chaque mot à ajouter, le système demande de taper "C" pour continuer à ajouter ou "A" pour arrêter d'ajouter.

Vous trouverez les spécifications et le code Pascal de cette procédure en annexe.

I. 5. 3. Suppression de mots.

Le système demande à l'utilisateur d'écrire le mot à supprimer en toutes lettres. S'il n'est pas dans le dictionnaire, il le dit puis demande à l'utilisateur s'il veut arrêter de supprimer ou continuer avec un autre mot.

S'il est dans le dictionnaire, il affiche les caractéristiques, soit pour les verbes leur numéro de classe, soit pour les mots leur genre, leurs possibilités de déterminant et de pluriel et leur(s) numéro(s) de classe. On demande alors de confirmer la suppression. Cela fait, le système demande de taper "C" pour continuer à supprimer ou "A" pour arrêter de supprimer. Vous trouverez les spécifications et le code Pascal de cette procédure en annexe.

I. 5. 4. Modification de mots.

Sur la moitié inférieure de l'écran est affichée la liste des classes de mots existantes.
Le système demande à l'utilisateur d'écrire le mot à modifier en toutes lettres. S'il n'est pas dans le dictionnaire, il le dit puis demande à l'utilisateur s'il veut arrêter de modifier ou continuer avec un autre mot. S'il est dans le dictionnaire, il affiche ses différentes caractéristiques. Puis, le curseur se positionne sur le mot. Si l'utilisateur tape "RETURN", l'orthographe n'est pas modifiée, sinon il doit d'abord taper la nouvelle orthographe. Il en est de même pour le genre, les possibilités de déterminant et de pluriel et la première classe. S'il y a plusieurs classes, l'utilisateur doit les écrire de toutesfaçons. Cela fait, le système demande à l'utilisateur de taper "C" pour continuer à modifier ou "A" pour arrêter de modifier.
Vous trouverez les spécifications et le code Pascal de cette procédure en annexe.

I. 5. 5. Consultation de mots.

Le système demande à l'utilisateur d'écrire le mot qu'il veut consulter. S'il n'est pas dans le dictionnaire, il le dit puis demande à l'utilisateur s'il veut continuer ou arrêter de consulter. S'il est dans le dictionnaire, il affiche ses différentes caractéristiques soit pour un verbe son numéro de classe, soit pour un mot son genre, ses possibilités de déterminant et de pluriel et sa ou ses numéro(s) de classe(s). Cela fait, le système demande à l'utilisateur de taper "C" pour continuer à consulter ou "A" pour arrêter de consulter.
Vous trouverez les spécifications et le code Pascal de cette procédure en annexe.

I. 5. 6. Liste de mots du dictionnaire.

L'utilisateur peut demander la liste des mots du dictionnaire à l'écran ou à l'imprimante. Les mots sont accompagnés de leurs caractéristiques. Ils sont classés par ordre alphabétique. Ces procédures permettent à l'utilisateur de visualiser le dictionnaire.
Vous en trouverez les spécifications et le code Pascal en annexe.

II. Gestion du modèle de sémantique.

La gestion du modèle de sémantique représente en fait la gestion du tableau en trois dimensions tel que décrit au paragraphe VIII du chapitre VI.

II. 1. Représentation du modèle de sémantique.

Le modèle de sémantique des phrases est représenté en mémoire centrale par un tableau de booléen en trois dimensions et en mémoire secondaire par un fichier de records.
Ces deux représentations sont décrites en détail aux paragraphes VIII et IX du chapitre VI.

II. 2. Gestion du tableau en trois dimensions.

II. 2. 1. Introduction.

L'enseignant doit pouvoir créer des modèles de phrases sémantiquement correctes et en supprimer.
Il doit aussi pouvoir les consulter. Ce sont ces trois procédures qui sont décrites ci-après.

II. 2. 2. Ajout de phrases.

Cette procédure permet à l'utilisateur de déterminer un modèle de phrases sémantiquement correctes "classe sujet - verbe - classe lieu".

Pour chaque modèle de phrase créé, le système demande à l'utilisateur d'écrire le numéro de la classe sujet tandis que sur la moitié inférieure de l'écran est affichée la liste des classes sujets existantes.
Ensuite, le module affiche la liste des verbes existants dans le dictionnaire et demande à l'utilisateur d'écrire le verbe en toutes lettres.
Enfin, le système demande à l'utilisateur d'écrire le numéro de la classe lieu tandis que sur la moitié inférieure de l'écran est affichée la liste des classes lieux existantes.
Entre chaque modèle de phrase à créer, le système demande à l'utilisateur de taper "C" pour continuer à créer ou "A" pour arrêter.
L'ajout d'un modèle de phrases correctes consiste simplement à mettre à vrai l'élément du tableau concerné.
Vous trouverez les spécifications et le code Pascal de cette procédure en annexe.

II. 2. 3. Suppression de phrases.

Cette procédure permet à l'utilisateur de supprimer un modèle de phrases sémantiquement correctes.
L'utilisation en est identique à celle d'ajout de phrases sauf qu'ici il s'agit de suppression.

La suppression d'un modèle de phrases correctes consiste simplement à mettre à faux l'élément du tableau concerné.
Vous trouverez les spécifications et le code Pascal de cette procédure également en annexe.

II. 2. 4. Liste des modèles de phrases correctes.

L'utilisateur peut demander la liste des modèles de phrases correctes à l'écran ou à l'imprimante.
Cette liste est visualisée de la façon suivante :
"Sujet" : "numéro de la classe" : "titre de la classe"
puis pour chaque sujet, on a la liste de ses verbes
"Verbe" : verbe écrit en toutes lettres
et pour chaque verbe, on a la liste de ses lieux
"Lieu" : "numéro de la classe" : "titre de la classe"

En voici un exemple :

sujet : 1 : humains

verbe : vivir

lieu : habitation
lieu : géographie humaine
lieu : noms propres
lieu : vêtements

verbe : trabajar

lieu : habitation
lieu : géographie humaine
:

Vous trouverez les spécifications et le code Pascal de ces deux procédures en annexe.

CONCLUSION.

Nous en arrivons donc à la conclusion de ce mémoire. La réalisation de ce dernier m'a permis d'approfondir mes connaissances :

comment travailler sur un micro-ordinateur.

comment réaliser un travail de programmation en tenant compte de contraintes de place mémoire disponible et de vitesse d'exécution.

comment mener à bien des recherches bibliographiques intenses (pour la correction d'orthographe et la génération de phrases), choses que j'avais déjà faites auparavant mais dans des proportions beaucoup moins importantes.

comment réaliser un travail de conception et de réalisation d'un projet informatique en équipe, avec tous les problèmes de communication d'informations que cela implique.

comment communiquer avec un client néophyte en informatique, en l'occurence Monsieur Régimont.

A cette liste, il faut également ajouter toute une série d'autres enseignements qui ne me viennent pas à l'esprit immédiatement mais qui me serviront certainement le moment venu dans la vie professionnelle.

Au moment d'écrire cette conclusion, un regret : celui de ne pas pouvoir ou avoir pu réaliser l'ensemble du didacticiel, et un espoir : celui que quelqu'un(ou quelques-uns) le fera(ont) et que les attentes de Monsieur Régimont ne seront pas déçues. J'ai pu constater que ce didacticiel, comme l'indique le titre de ce mémoire, est réellement faisable. On peut donc espérer que, à plus ou moins long terme, les élèves de Monsieur Régimont ne se tromperont plus dans l'emploi des prépositions "a" et "en".

TABLE DES MATIERES.

BUMP
0 0 3 7 4 4 5 3 5
*FM B16/1984/37/1

Institut d'Informatique
21, Rue Grandgagnage
B-5000 NAMUR

ANNEE ACADEMIQUE 1983-84

Etude de faisabilité d'un didacticiel d'acquisition de structures morphosyntaxiques espagnoles.

****

Annexes.

Promoteur : Monsieur Claude CHERTON

Jacques KINET

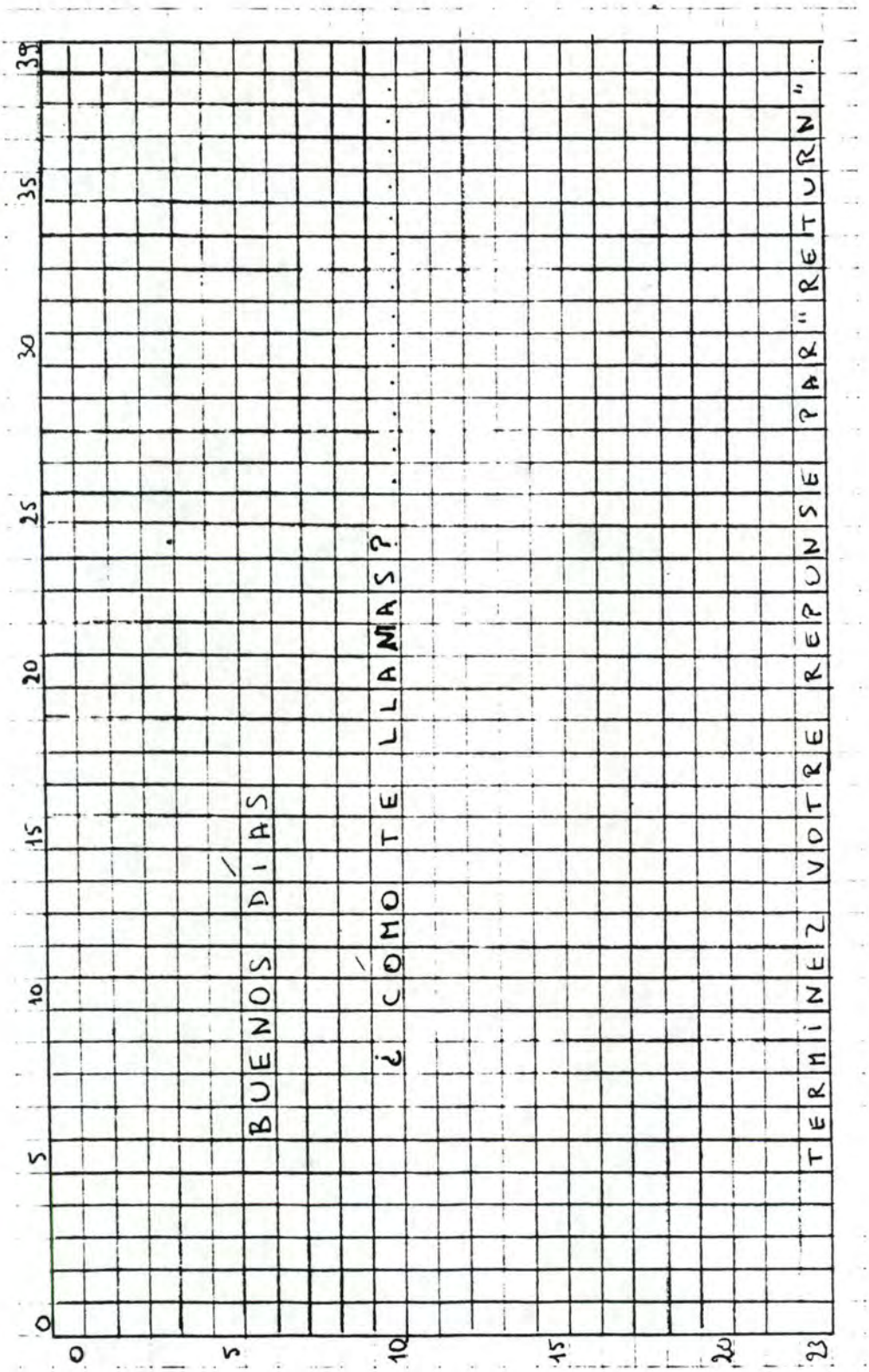
Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

PLAN DES ANNEXES.

A N N E X E 1.

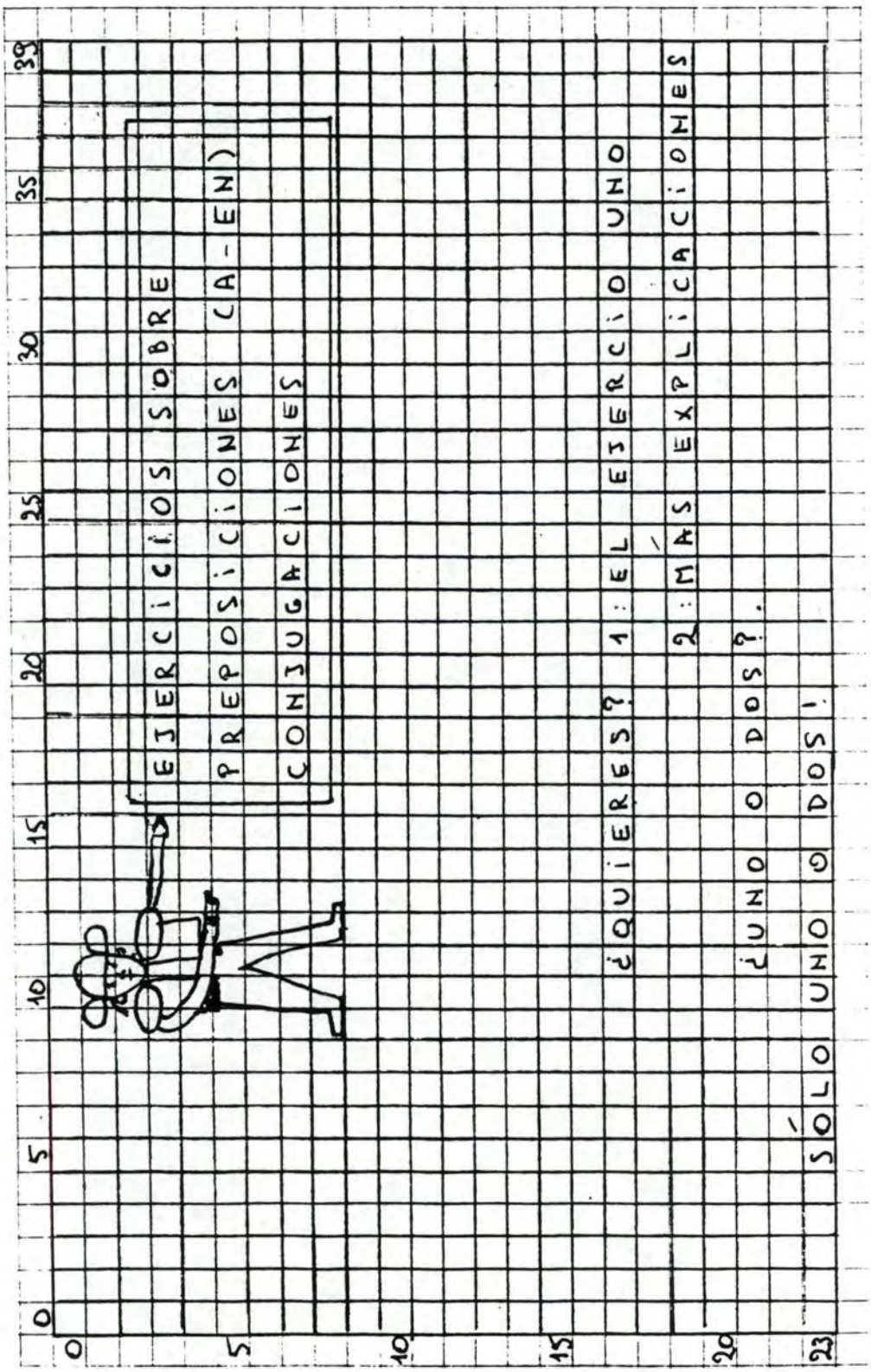
GRILLES D'ECRAN DU DIDACTICIEL.

Ecran 1 : Accueil.



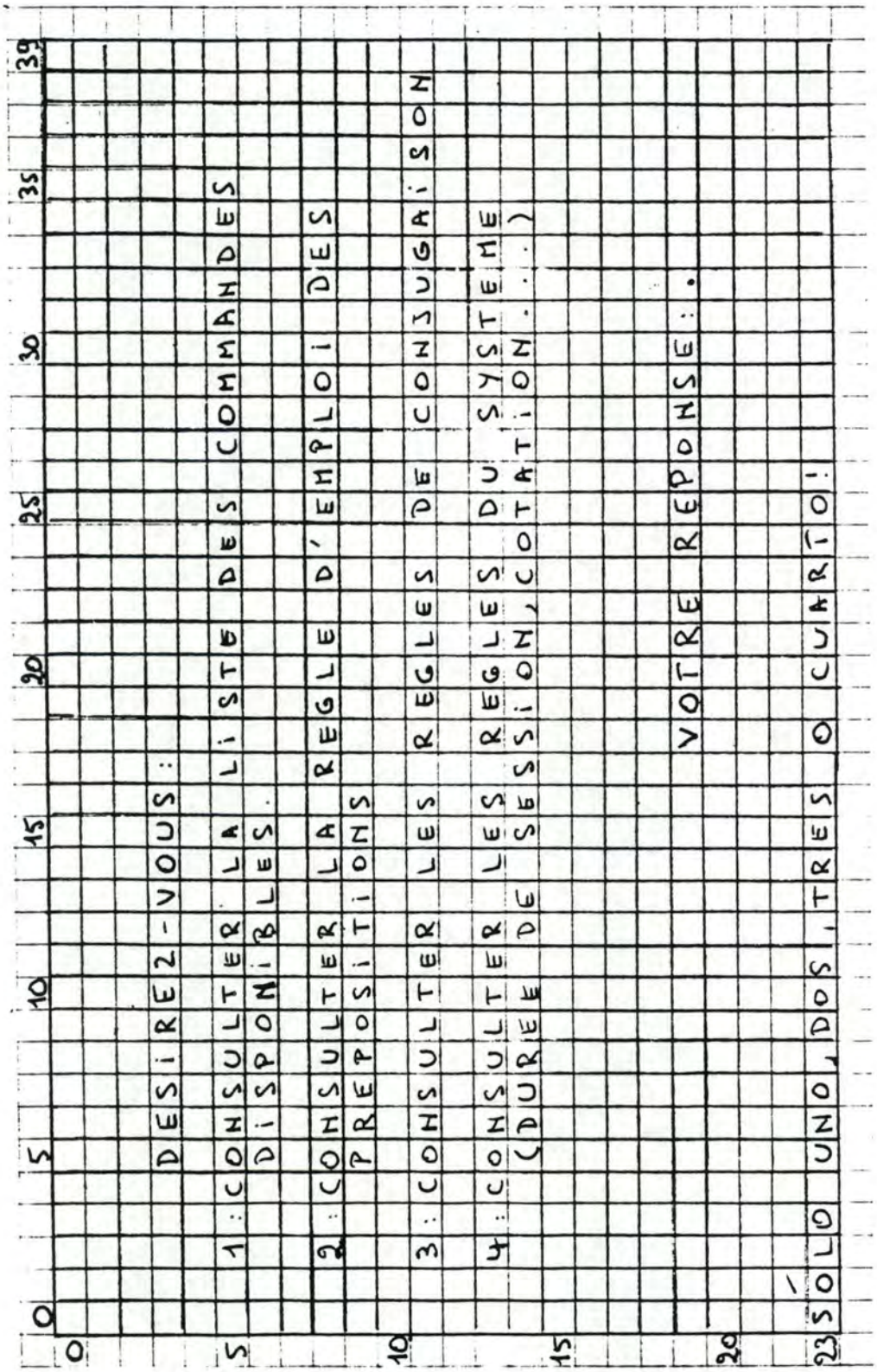
Il s'agit de l'écran que l'utilisateur rencontre quand il veut se servir du didacticiel. Il l'accueille et lui demande son nom. Si le système accepte celui-ci, l'écran 2 apparaîtra. Dans le cas contraire, l'utilisateur n'ira pas plus loin et la session sera terminée avant même d'avoir réellement commencé.

Ecran 2 : Introduction au système.



Par l'intermédiaire de cet écran, l'utilisateur accède au système en venant de l'écran 1. On lui propose soit de commencer le premier exercice, soit de demander plus d'explications.
"Sólo uno o dos" apparaît uniquement lorsqu'il tape autre chose que 1 ou 2.
S'il tape 1, l'écran d'introduction aux premiers exercices de sa session apparaît.
S'il tape 2, l'écran 3 de menu d'aide apparaît.

Ecran 3 : Menu d'aide.



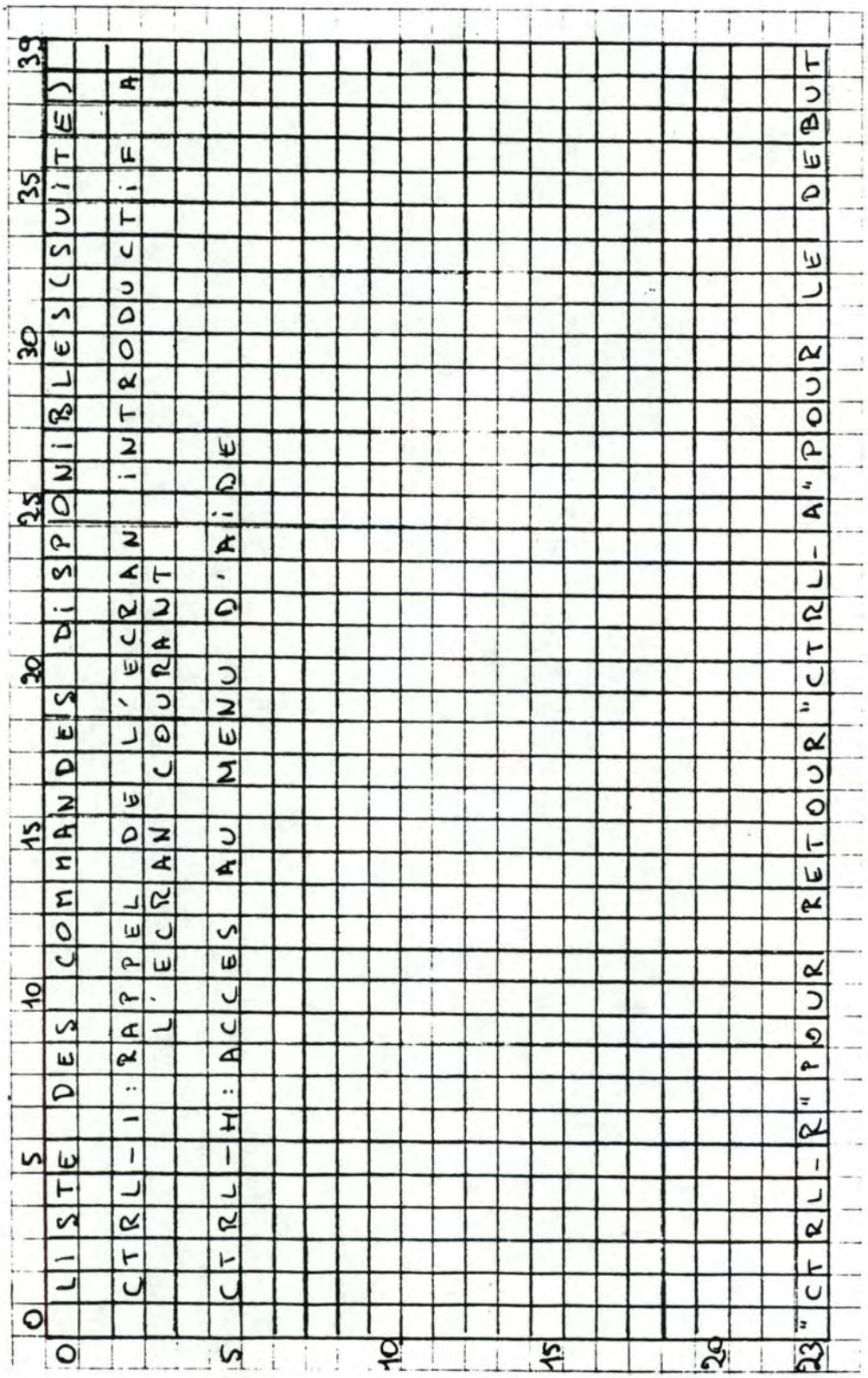
Cet écran apparaît lorsque l'utilisateur a demandé de l'aide pendant la session ou lorsqu'il a demandé plus d'explications à l'écran 2.
Il donne à choisir entre quatre sujets d'information.
"Sólo uno, dos, tres o cuarto !" apparaît uniquement lorsque l'utilisateur tape autre chose que "1", "2", "3" ou "4".
S'il tape "1", l'écran 4 : liste des commandes disponibles apparaît.
S'il tape "2", l'écran 5 : règle d'emploi des prépositions apparaît.
S'il tape "3", l'écran 6 : règles de conjugaison apparaît.
S'il tape "4", l'écran 7 : règles du système apparaît.

Ecran 4 : Liste des commandes disponibles:

```
LISTE DES COMMANDES DISPONIBLES

RETURN: TERMINE CHAQUE REPONSE

<---: DEPLACEMENT DU CURSEUR VERS LA GAU-
     CHE SANS MODIFIER CE QUi EST ECRiT

--->: DEPLACEMENT DU CURSEUR VERL LA DROi
     TE SANS MODiFiER CE QUi EST ECRiT

BARRE D'ESPACEMENT: REMPLACEMENT DU CA-
     RACTERE DESiGNE PAR LE CURSEUR
     PAR UN ESPACE. (SUPPRESSION)

MODiFiER UN CARACTERE: POSiTiONNER LE
     CURSEUR SUR LE CARACTERE A MODi-
     FIER PUiS TAPER LE CARACTERE QUE
     VOUS VOULEZ ECRiRE A CETTE PLACE.

CTRL-O: REMONTER DANS L'HiSTORiQUE DES
     REPONSES PASSEES

CTRL-R POUR RETOUR CTRL-A POUR LA SUiTE
```

L'utilisateur accède à cet écran par l'intermédiaire de l'écran 3 : menu d'aide. Cet écran permet à l'utilisateur de prendre connaissance de la liste des commandes du système lui permettant de répondre aux questions et d'accomplir les exercices demandés.
Comme les dimensions de l'écran ne sont pas suffisantes pour afficher toutes les informations, celles-ci sont disséminées sur deux écrans différents, l'écran 4 et l'écran 4 bis.
Lorsque l'utilisateur tape "CTRL-R", l'écran d'où il était parti pour accéder au menu d'aide lui apparaît.
S'il tape "CTRL-A", il accède à l'écran 4 bis : liste des commandes disponibles (suite).

Ecran 4 bis : Liste des commandes disponibles (suite).

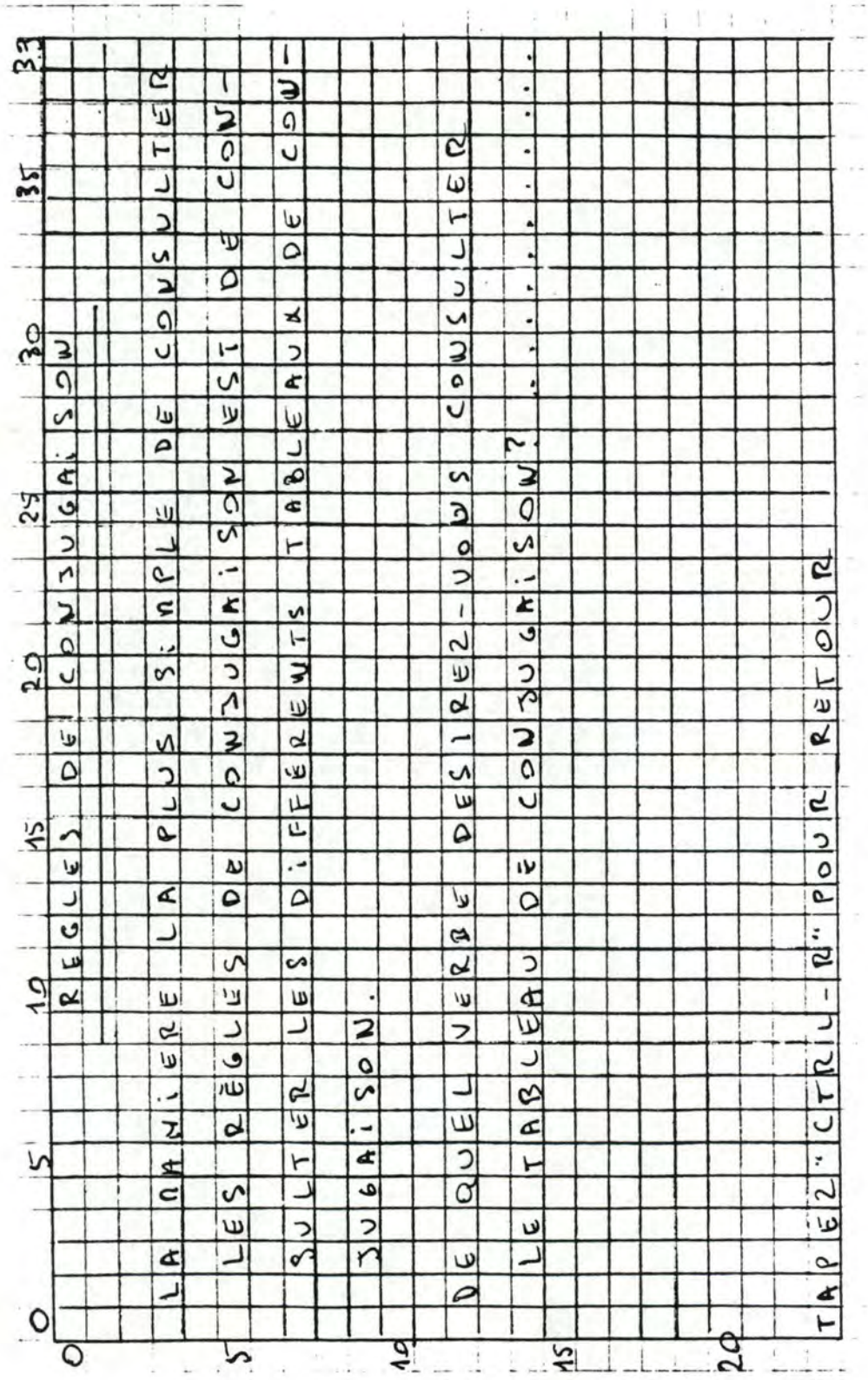


L'utilisateur accède à cet écran par l'intermédiaire de l'écran 4 : liste des commandes disponibles.
Cet écran permet à l'utilisateur de prendre connaissance des commandes du système qui ne seraient pas sur l'écran 4.
Lorsqu'il tape "CTRL-R", l'écran d'où il était parti avant d'accéder au menu d'aide (écran 3) puis à la liste des commandes disponibles (écran 4) lui apparaît.
S'il tape "CTRL-A", il accède à l'écran 4 : liste des commandes disponibles.

Ecran 5 : Règles d'emploi des prépositions.

L'utilisateur accède à cet écran par l'intermédiaire de l'écran 3 : menu d'aide. Il lui permet de consulter les régles d'emploi des prépositions de lieu "A" et "EN".
Lorsque l'utilisateur tape "CTRL-R", l'écran d'où il était parti pour accéder au menu d'aide lui apparaît.

Ecran 6 : Règles de conjugaison.



L'utilisateur accède à cet écran par l'intermédiaire de l'écran 3 : menu d'aide. Cet écran lui permet de choisir le verbe dont il veut consulter les tableaux de conjugaison :

- pour les verbes réguliers terminés par "ar" : l'écran 6a des tableaux de conjugaison de "cantar"
- pour les verbes réguliers terminés par "er" : l'écran 6b des tableaux de conjugaison de "beber"
- pour les verbes réguliers terminés par "ir" : l'écran 6c des tableaux de conjugaison de "vivir"
- pour l'auxiliaire "ser", l'écran 6d de ses tableaux de conjugaison
- pour le verbe "estar", l'écran 6e de ses tableaux de conjugaison
- pour l'auxiliaire "haber", l'écran 6f de ses tableaux de conjugaison
- pour les verbes "marchar(se)" et "quedar(se)", l'écran 6g des tableaux de conjugaison de "situar(se)
- pour le verbe "llegar", l'écran 6h de ses tableaux de conjugaison
- pour le verbe "morir", l'écran 6i de ses tableaux de conjugaison
- pour le verbe "ir", l'écran 6j de ses tableaux de conjugaison
-pour le verbe "venir", l'écran 6k de ses tableaux de conjugaison.

Lorsqu'il consulte des tableaux de conjugaison, l'utilisateur a la possibilité de positionner le curseur sur la forme conjuguée qui l'intéresse. Cela a pour effet d'afficher dans le coin de l'écran de retour cette forme conjuguée.

Lorsque l'utilisateur tape "CTRL-R", l'écran d'où il était parti pour accéder au menu d'aide lui réapparaît; c'est l'écran de retour où reste affichée la forme conjuguée désignée par le curseur.

Les écrans 6a à 6k reprennent les tableaux de conjugaison détaillés au point 5 du chapitre I du mémoire. Ils contiennent également les formes à l'infinitif des verbes employés par le didacticiel et se conjuguant comme le modèle du tableau concerné.

Les grilles des écrans 6a à 6k n'ont pas été réalisées.

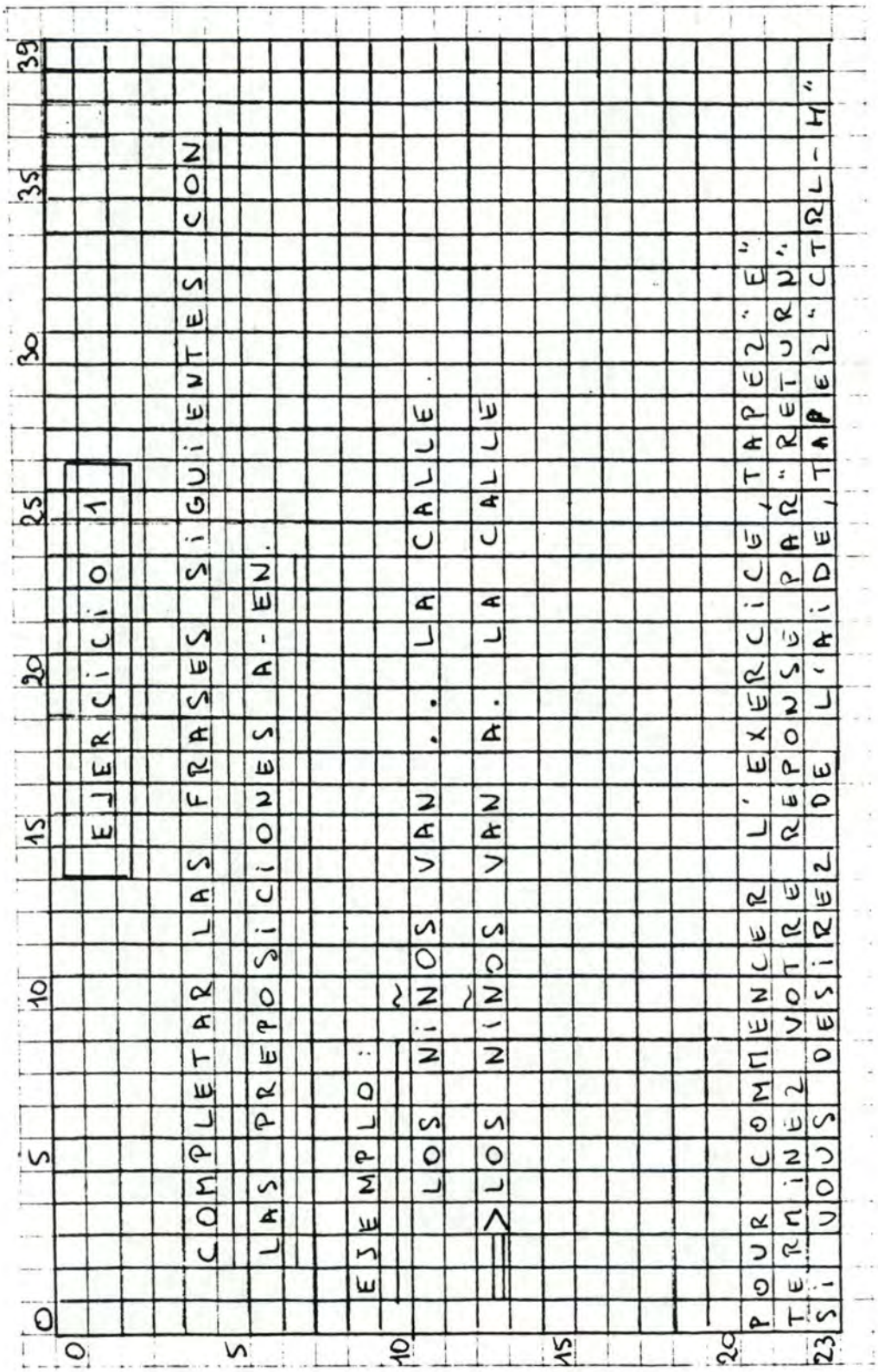
Ecran 7 : Règles du système.

La grille de cet écran n'a pas été réalisée.

L'utilisateur accède à cet écran par l'intermédiaire de l'écran 3 : menu d'aide. Cet écran lui permet de prendre connaissance des règles du système en ce qui concerne la durée de la session, la cotation, le degré de difficulté des exercices, etc...

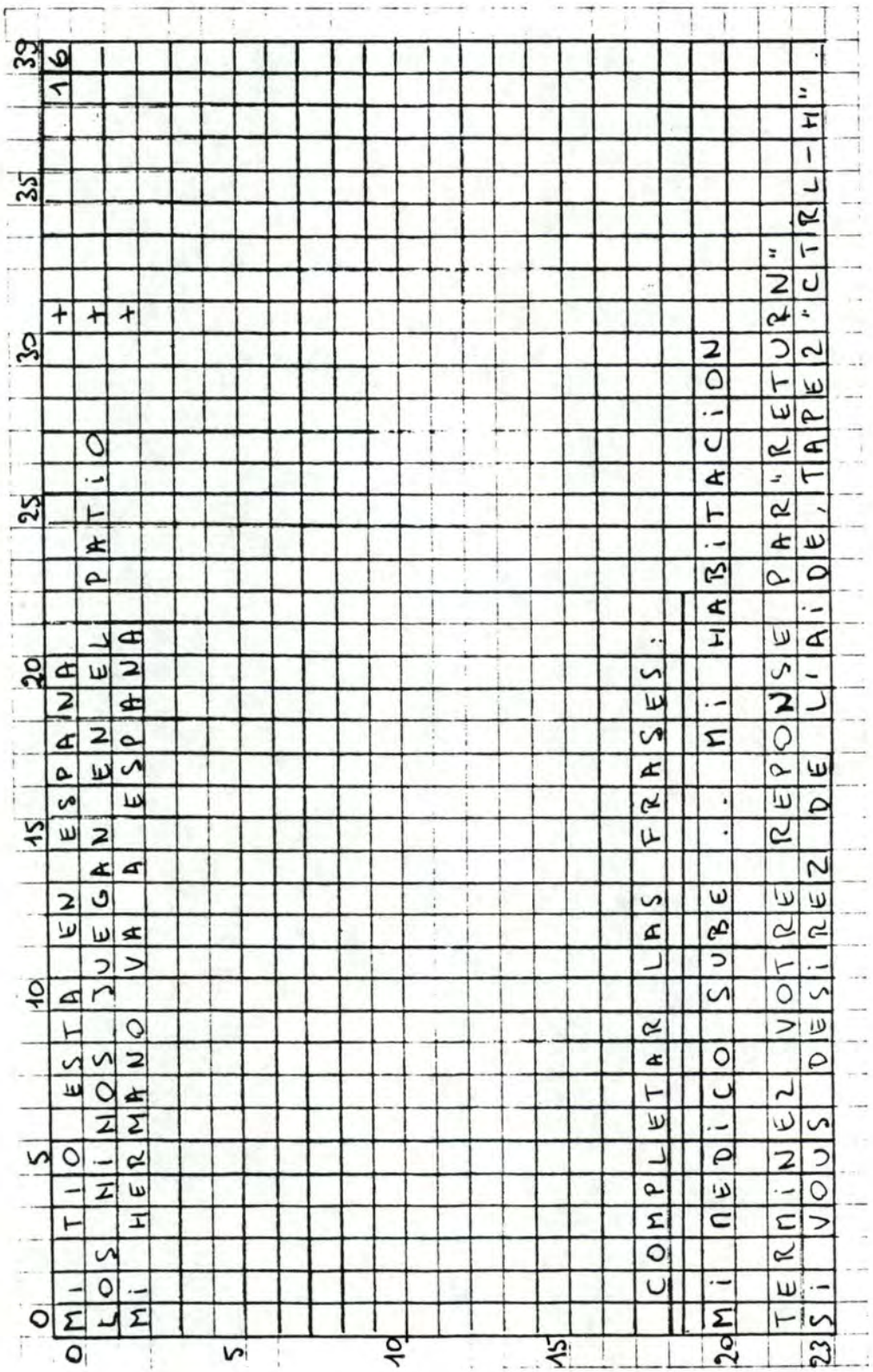
Lorsque l'utilisateur tape "CTRL-R", l'écran d'où il était parti pour accéder au menu d'aide lui réapparaît.

Ecran 8 : Introduction au type 1.



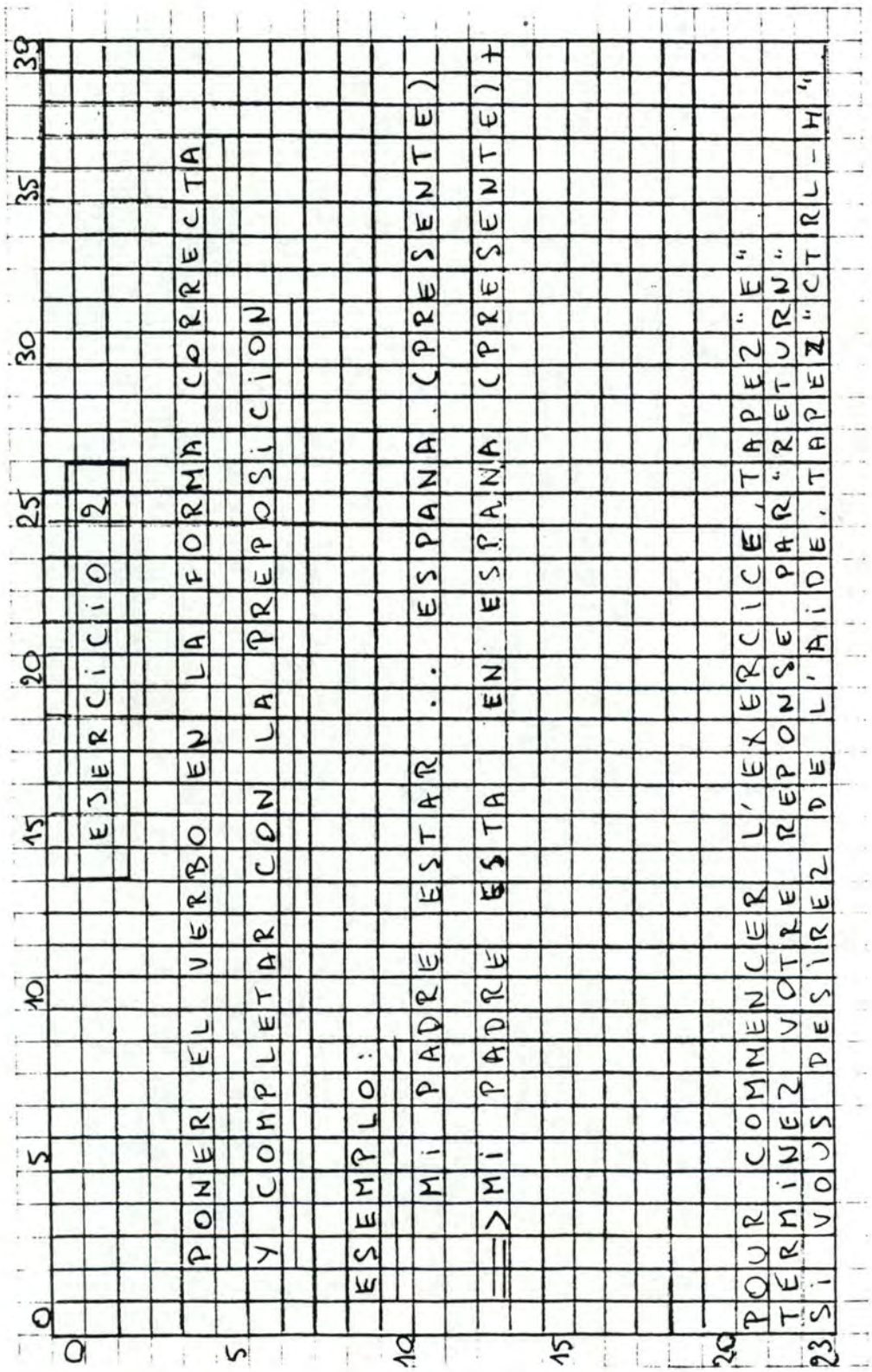
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 1 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice de type 1 ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 9 : exercices de type 1.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 9 : Exercices de type 1.



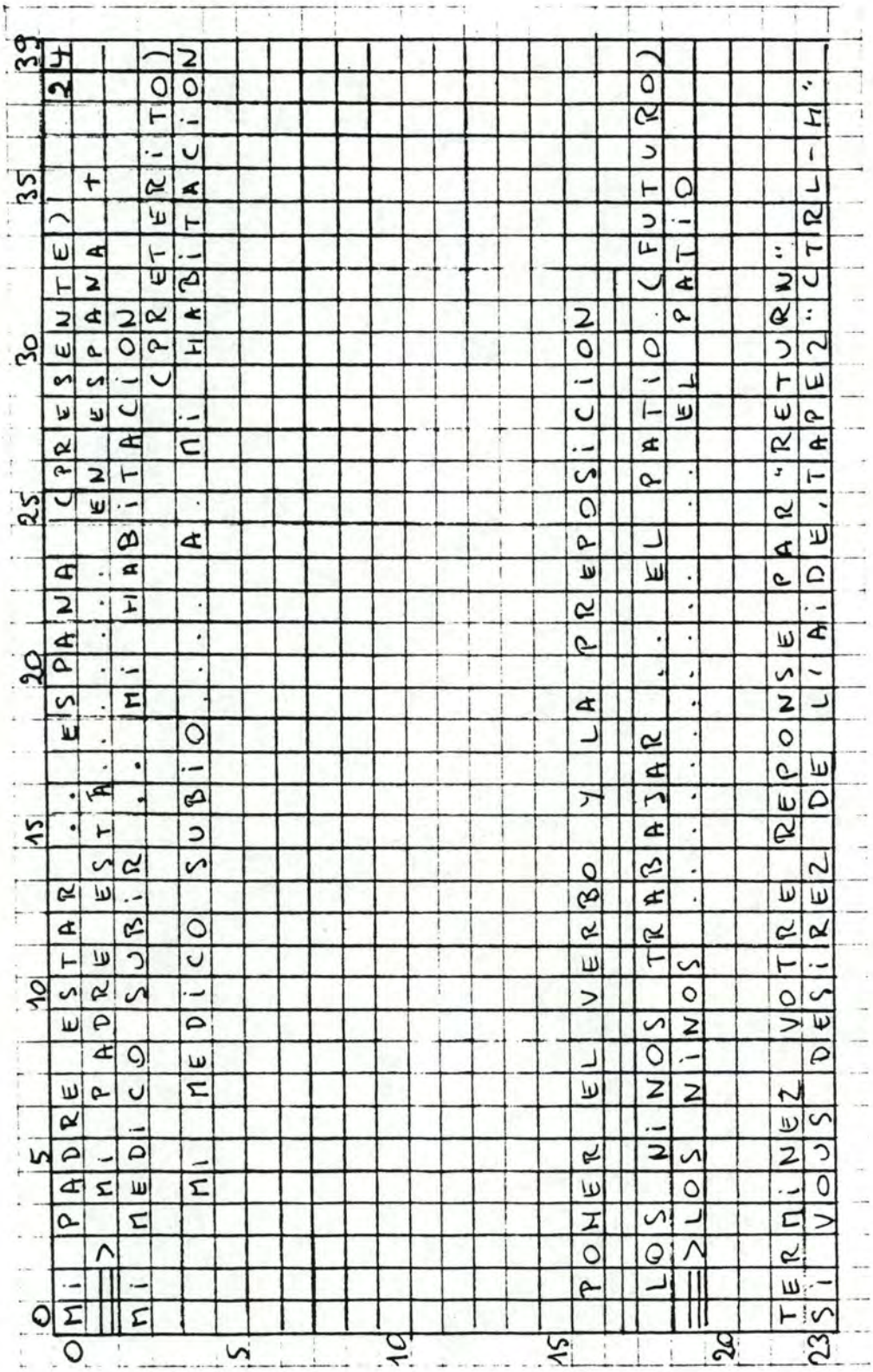
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 8 : introduction au type 1. Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 1. Sa cote est affichée en permanence dans le coin supérieur droit de l'écran et elle est maintenue à jour en permanence à chaque modification.
La phrase exercice courante apparaît en-dessous de l'intitulé de l'exercice au bas de l'écran. L'utilisateur fait l'exercice. Quand il a fini, la phrase vient occuper la première position libre en haut de l'écran au-dessus de l'intitulé de l'exercice et sous les phrases précédentes s'il y en a. Si cet espace est rempli, tout le haut de l'écran, reprenant l'historique des réponses passées, défile vers le haut pour libérer une place permettant à la dernière phrase de s'afficher au-dessus de l'intitulé de l'exercice. Lorsque l'utilisateur se trompe dans sa réponse, l'écran 29 : correction des réponses lui apparaît. S'il tape "CTRL-H", l'écran 3 : menu d'aide lui apparaît. Lorsque l'utilisateur a résolu toute la série des exercices de type 1 qui lui a été soumise, le système lui propose de continuer en tapant "CTRL-E".
Après cela, si la session d'exercices de l'utilisateur comporte encore d'autre(s) type(s) d'exercices, l'écran d'introduction au type d'exercice suivant apparaît.
Dans le cas contraire, c'est l'écran 28 : sortie du système qui s'affichera.

Ecran 10 : Introduction au type 2.



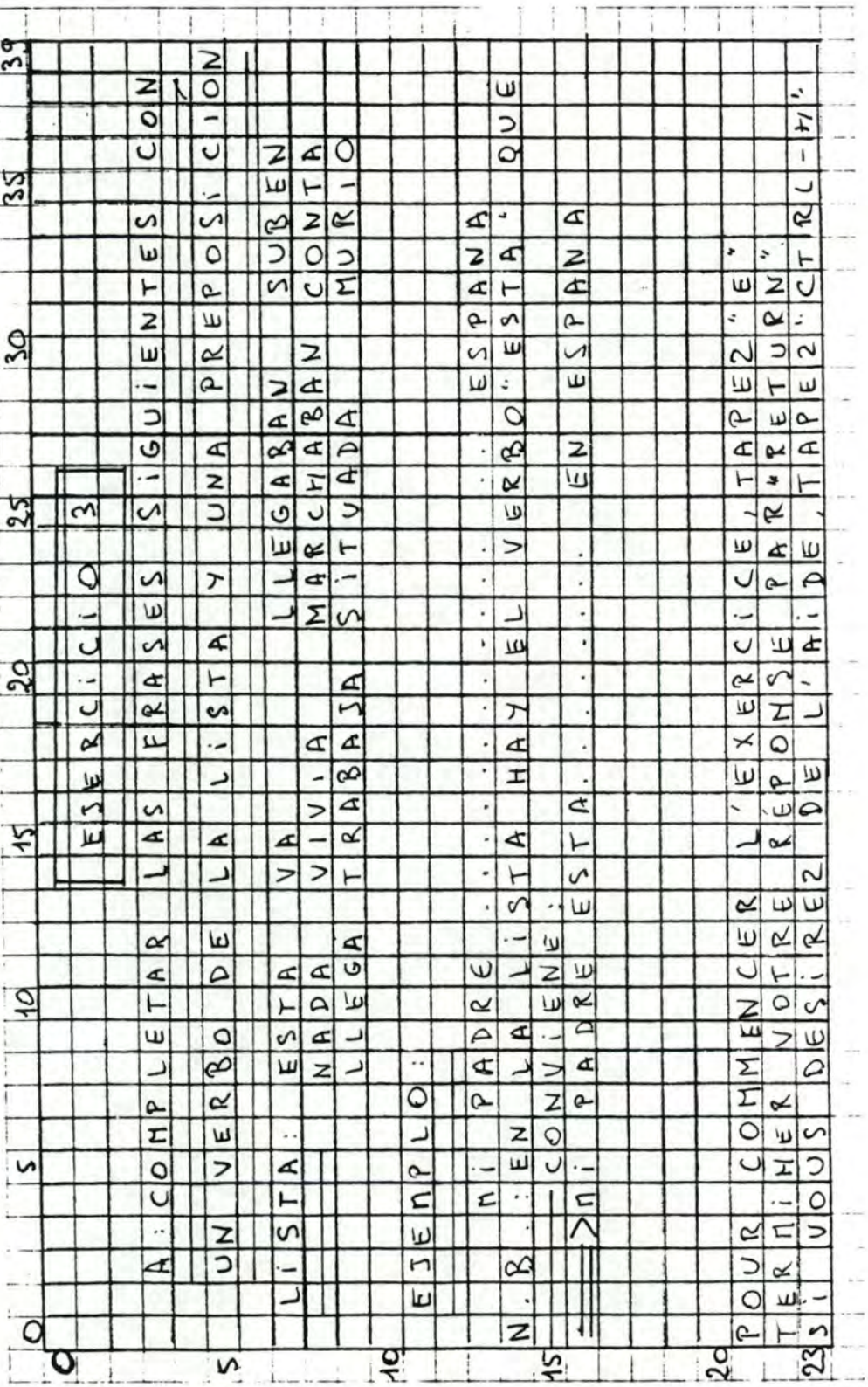
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accèdera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 1 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice de type 2 ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 11 : exercices de type 2.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 11 : Exercices de type 2.



L'utilisateur accède à cet écran par l'intermédiaire de l'écran 10. : introduction au type 2.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 2.
Sa description est similaire à celle de l'écran 9. : exercices de type 1.

Ecran 12. Introduction au type 3 a.

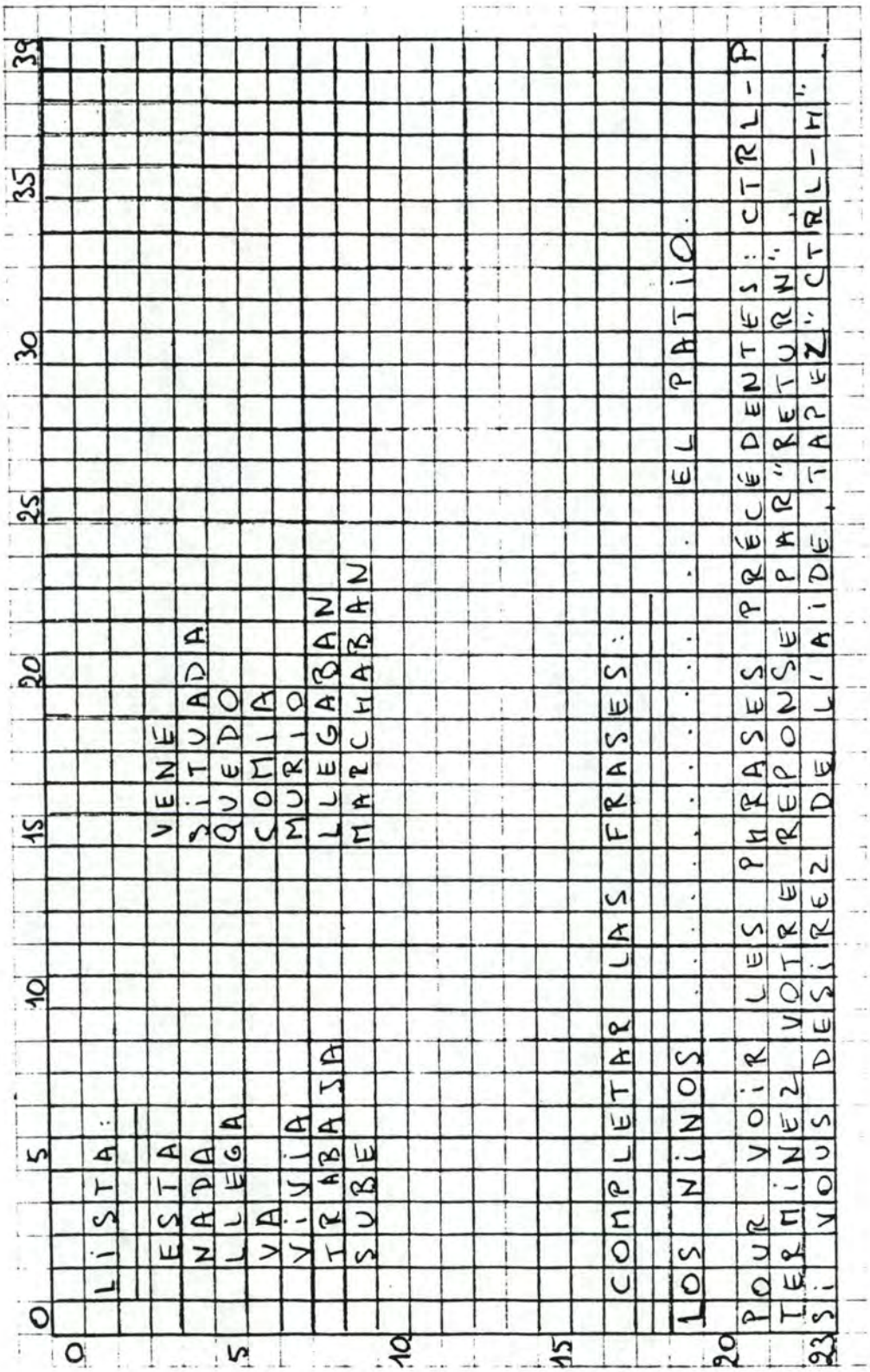


Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 1 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice de type 3 a, de la liste des verbes conjugués qui lui sont proposés ainsi que d'un exemple de résolution d'un exercice.
Lorsque l'utilisateur tape "E", il accède à l'écran 13 : exercices de type 3 a sans liste de verbes.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 13 : Exercices de type 2a sans liste de verbes.

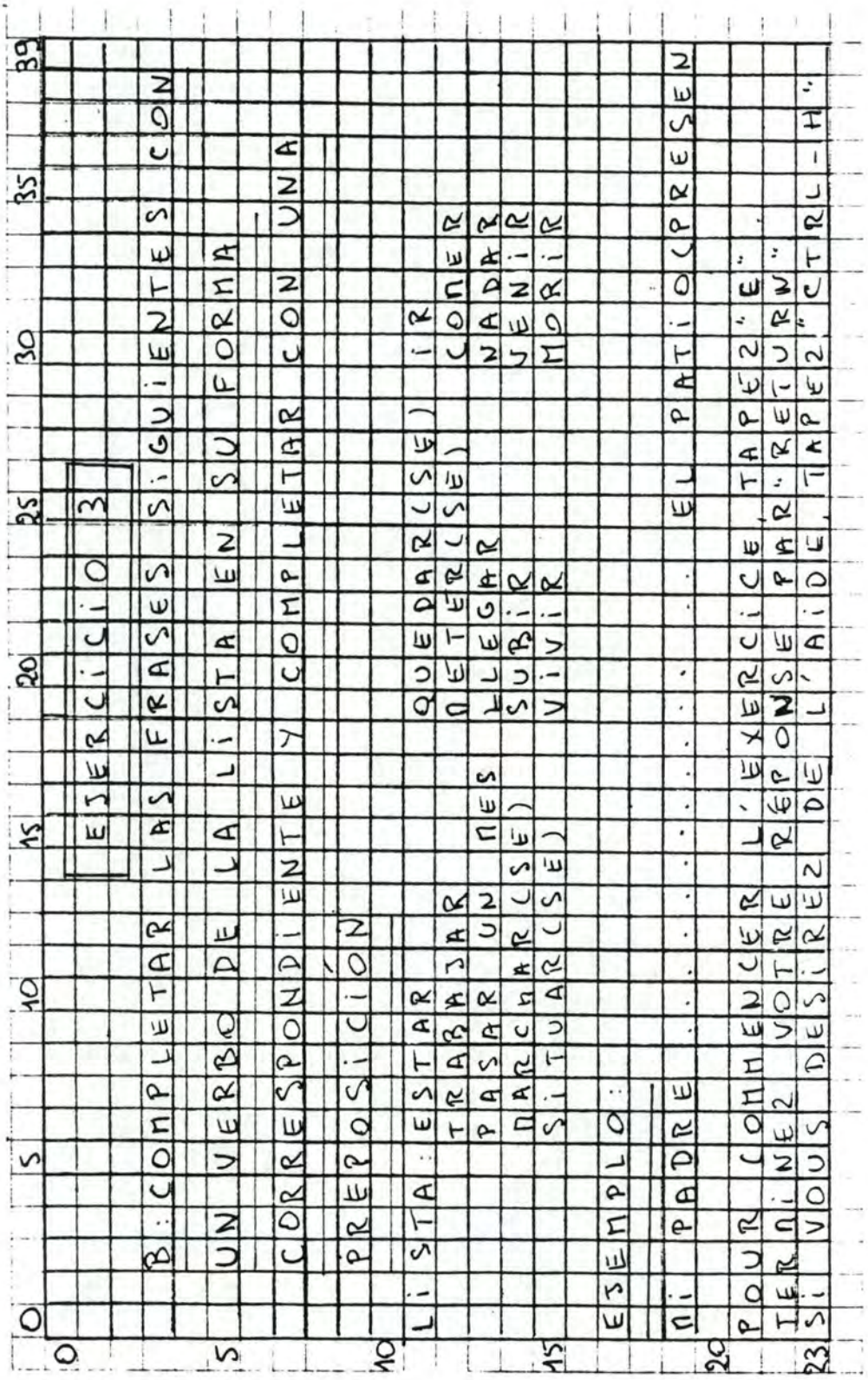
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 12 : introduction au type 3a.
Il peut également y revenir après avoir demandé de l'aide ou après être passé à l'écran 14.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 3a.
Sa description est similaire à celle de l'écran 9 : exercices de type 1.
Toutefois, lorsque l'utilisateur tape "CTRL-L", l'écran 14 : exercices de type 3a avec liste de verbes apparaît.
Lorsque l'utilisateur a résolu toute la série des exercices de type 3a qui lui a été soumise, le système lui propose de continuer en tapant "CTRL-E".
Après cela, si la session d'exercices de l'utilisateur comporte encore d'autre(s) type(s) d'exercice(s), l'écran d'introduction au type d'exercice suivant apparaîtra.
Dans le cas contraire, c'est l'écran 28 : Sortie du système qui s'affichera.

Ecran 14 : Exercices de type 3 a avec liste de verbes.



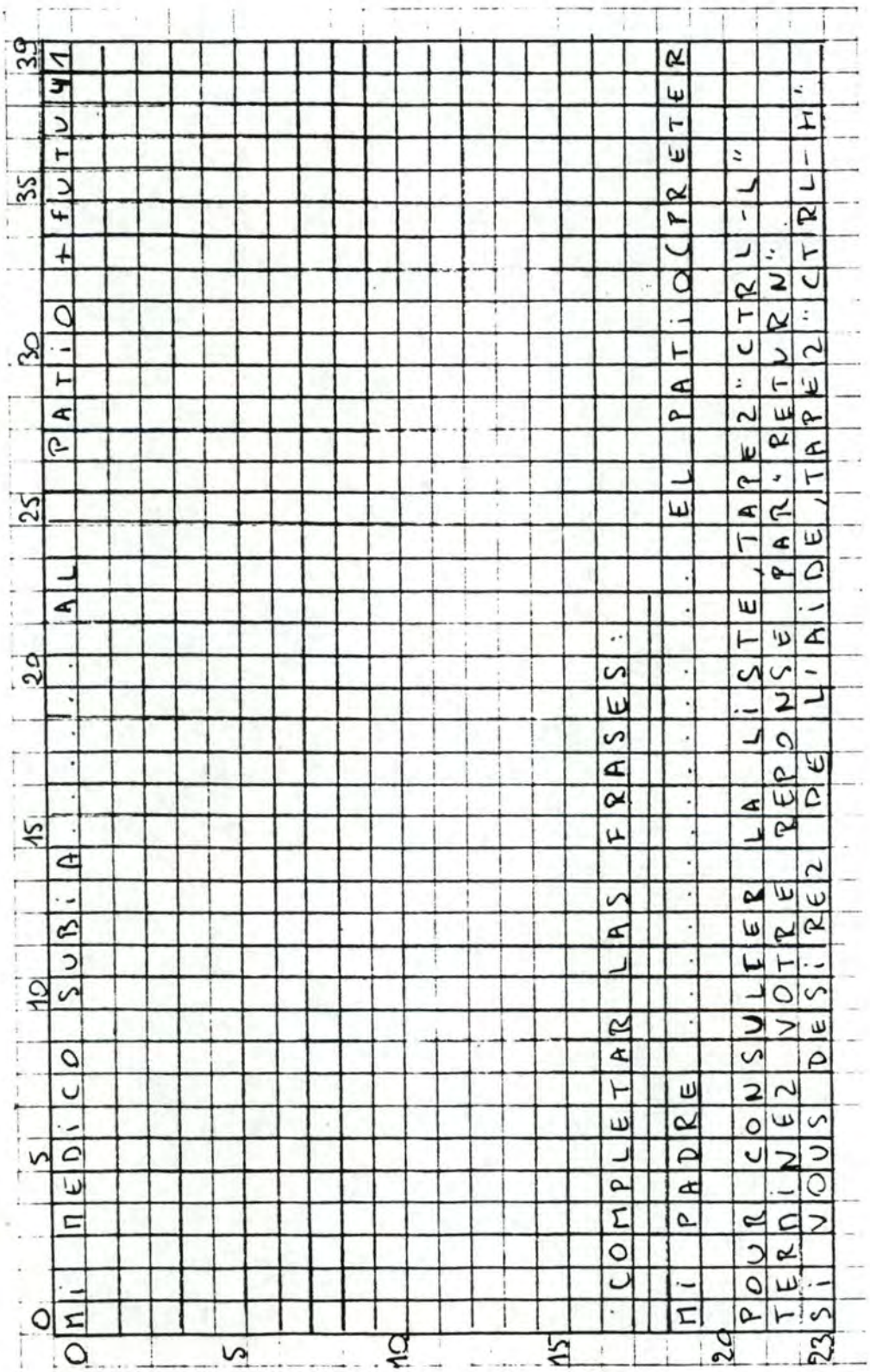
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 13 : exercices de type 3 a sans liste de verbes. Il peut également y revenir après avoir demandé de l'aide. Cet écran permet à l'utilisateur d'accomplir les exercices de type 3 a. Sa cote est affichée en permanence dans le coin supérieur droit de l'écran et elle est maintenue à jour en permanence à chaque modification.
La phrase-exercice courante apparaît en-dessous de l'intitulé de l'exercice en bas de l'écran. Dans le haut de ce dernier apparaît la liste des verbes conjugués.
Quand l'élève a fait une phrase-exercice, celle-ci disparaît pour faire place à la suivante.
Lorsque l'utilisateur se trompe dans sa réponse, l'écran 28 : correction des réponses lui apparaît.
S'il tape "CTRL-H", c'est l'écran 3 : menu d'aide qui apparaît et s'il tape "CTRL-P", c'est l'écran 12 : exercices de type 3 a sans liste de verbes qui s'affiche.
Lorsque l'utilisateur a résolu toute la série des exercices de type 3 a qui lui a été soumise, le système lui propose de continuer en tapant "CTRL-E". Après cela, si la session d'exercices de l'utilisateur comporte encore d'autre(s) type(s) d'exercice(s), l'écran d'introduction au type d'exercice suivant apparaîtra. Dans le cas contraire,c'est l'écran 28 : sortie du système qui s'affichera.

Ecran 15 : Introduction au type 3 b.



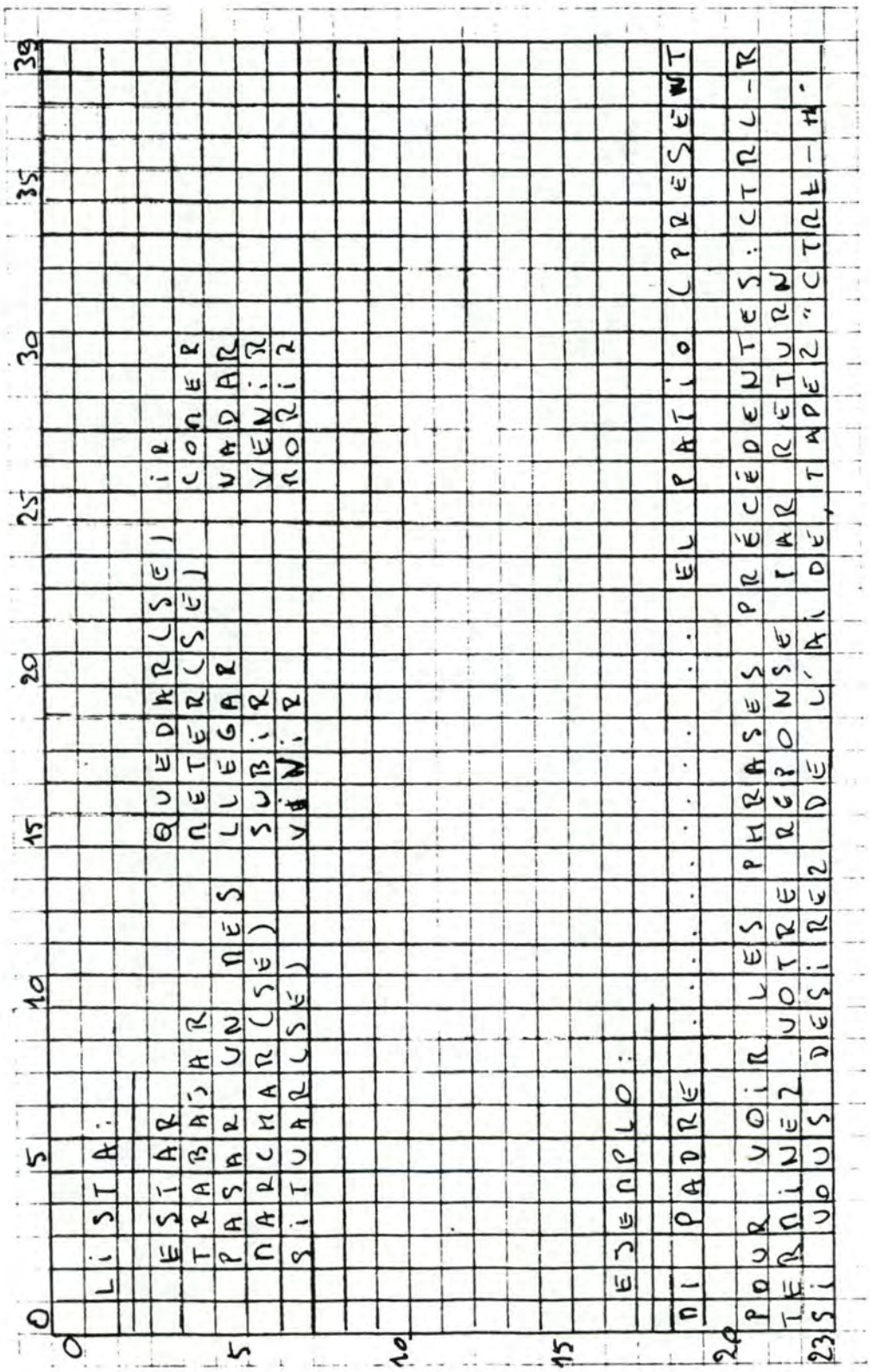
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 2 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice de type 3 b, de la liste des verbes à l'infinitif qui lui sont proposés ainsi que d'un exemple de résolution d'un exercice.
Lorsque l'utilisateur tape "E", il accède à l'écran 16 : exercices de type 3 b sans liste de verbes.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 16 : Exercices de type 3 b sans liste de verbes.



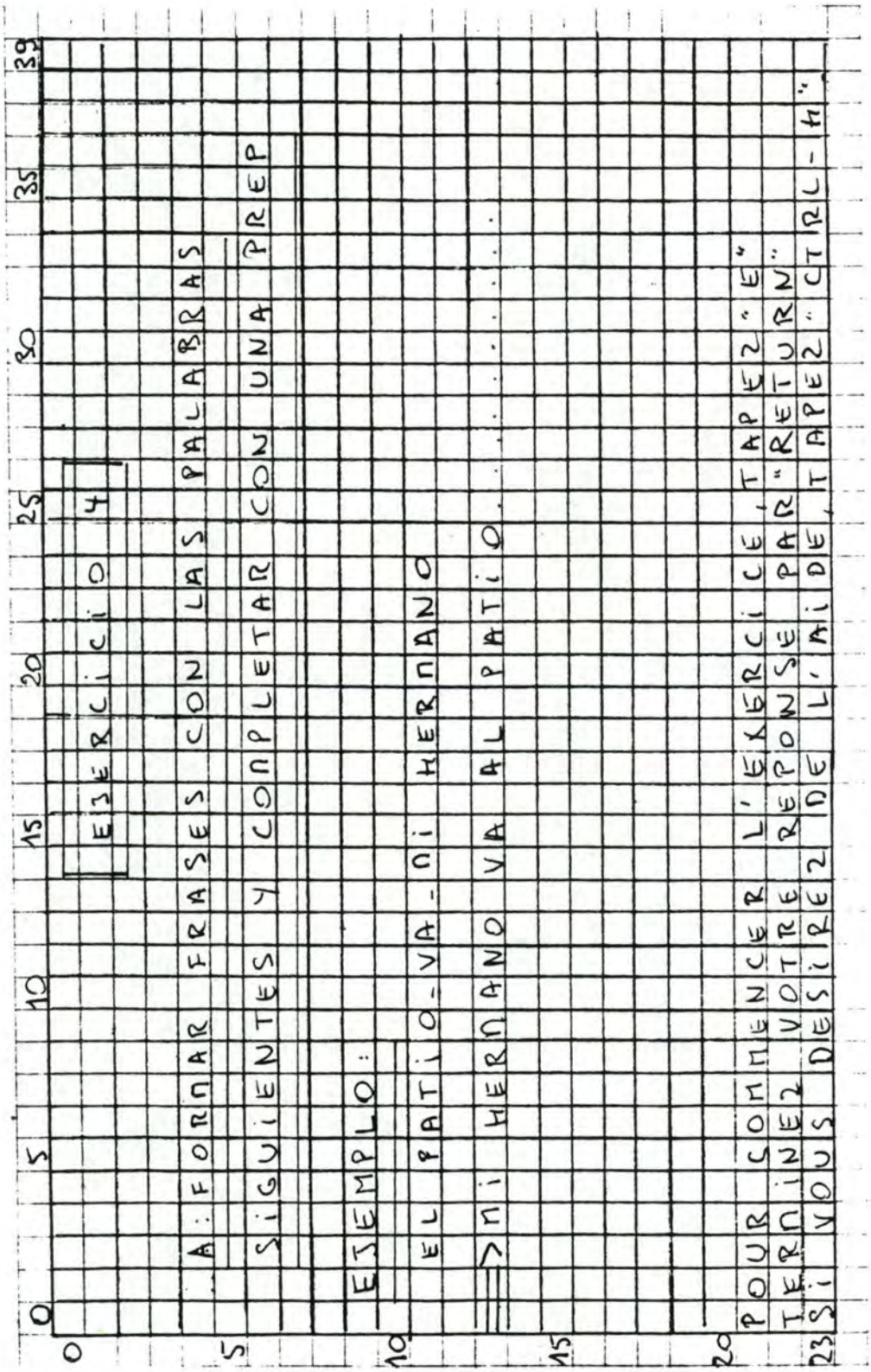
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 14 : introduction au type 3 b.
Il peut également y revenir après avoir demandé de l'aide ou après être passé à l'écran 15 : exercices de type 4 b avec liste de verbes.
Sa description est similaire à celle de l'écran 13 : exercices de type 3 a sans liste de verbes.

Ecran 17 : Exercices de type 3b avec liste de verbes.



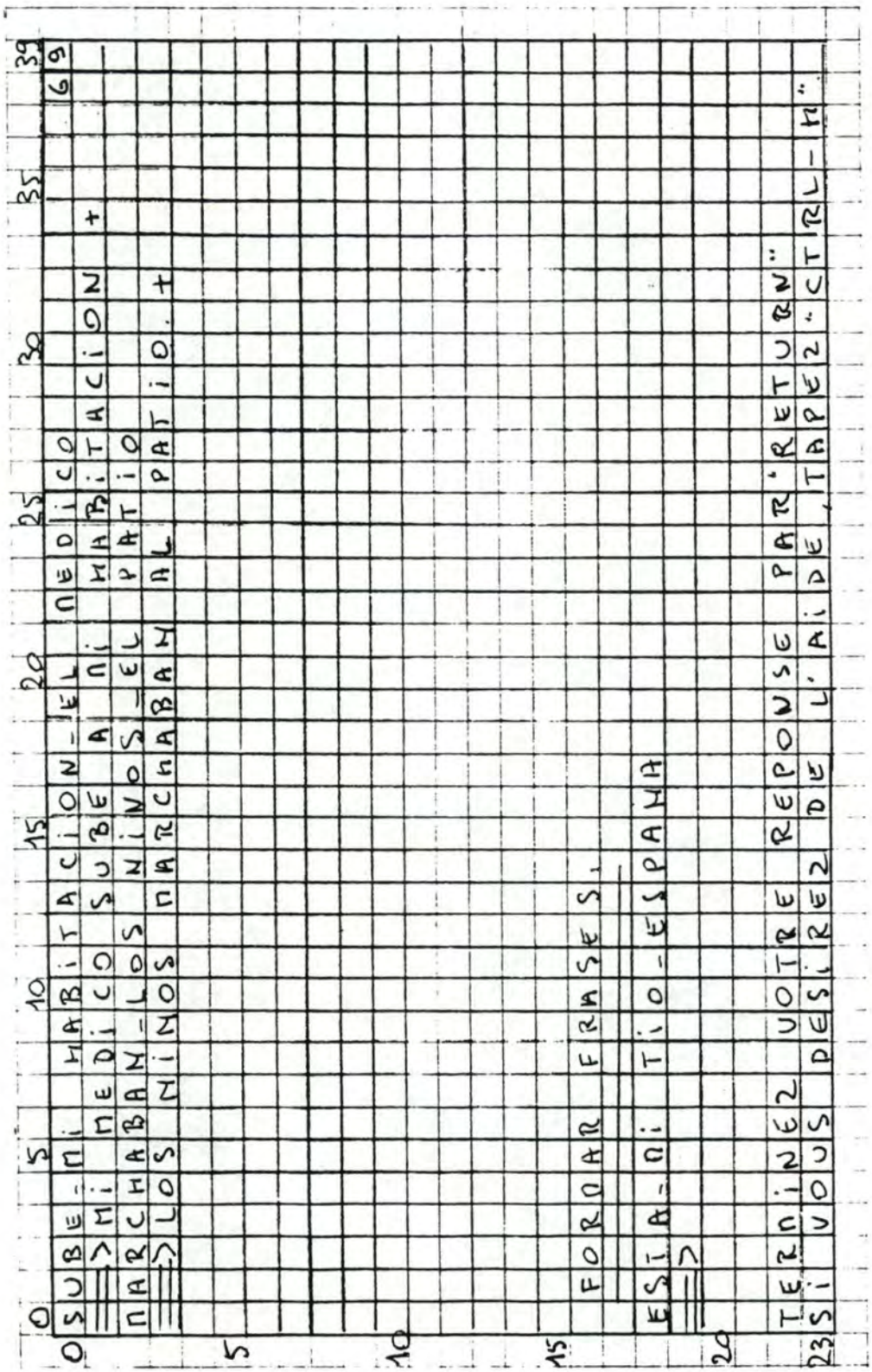
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 16 : exercices de type 3b sans liste de verbes.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 3b.
Sa description est similaire à celle de l'écran 14 : exercices de type 3a avec liste de verbes.
Si l'utilisateur tape "CTRL-H", c'est l'écran 3 : menu d'aide qui apparaît et s'il tape "CTRL-P", c'est l'écran 16 : exercices de type 3b sans liste de verbes qui s'affiche.

Ecran 18 : Introduction au type 4 a.



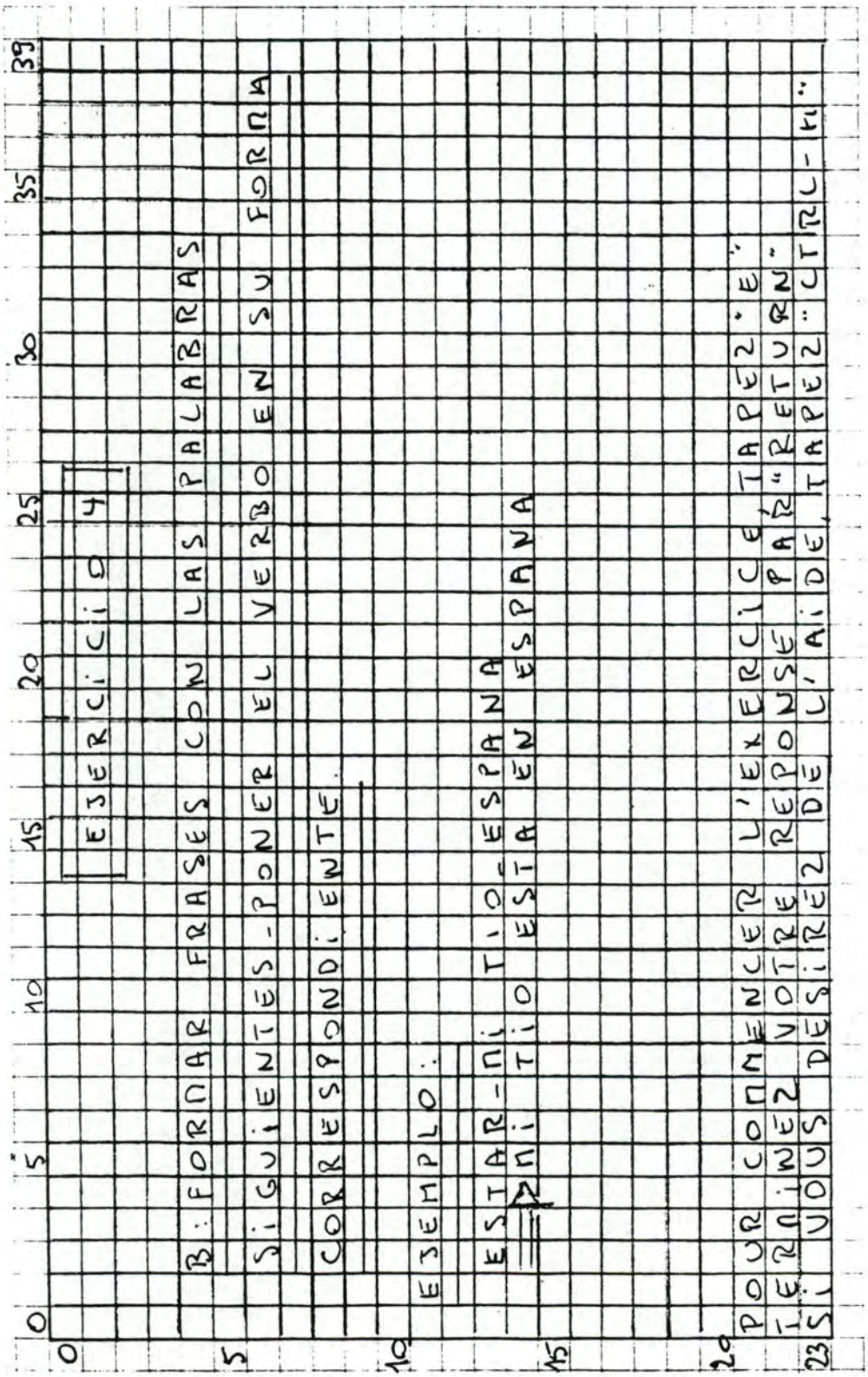
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 2 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 19 : exercices de type 4 a.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 19 : Exercices de type 4 a.



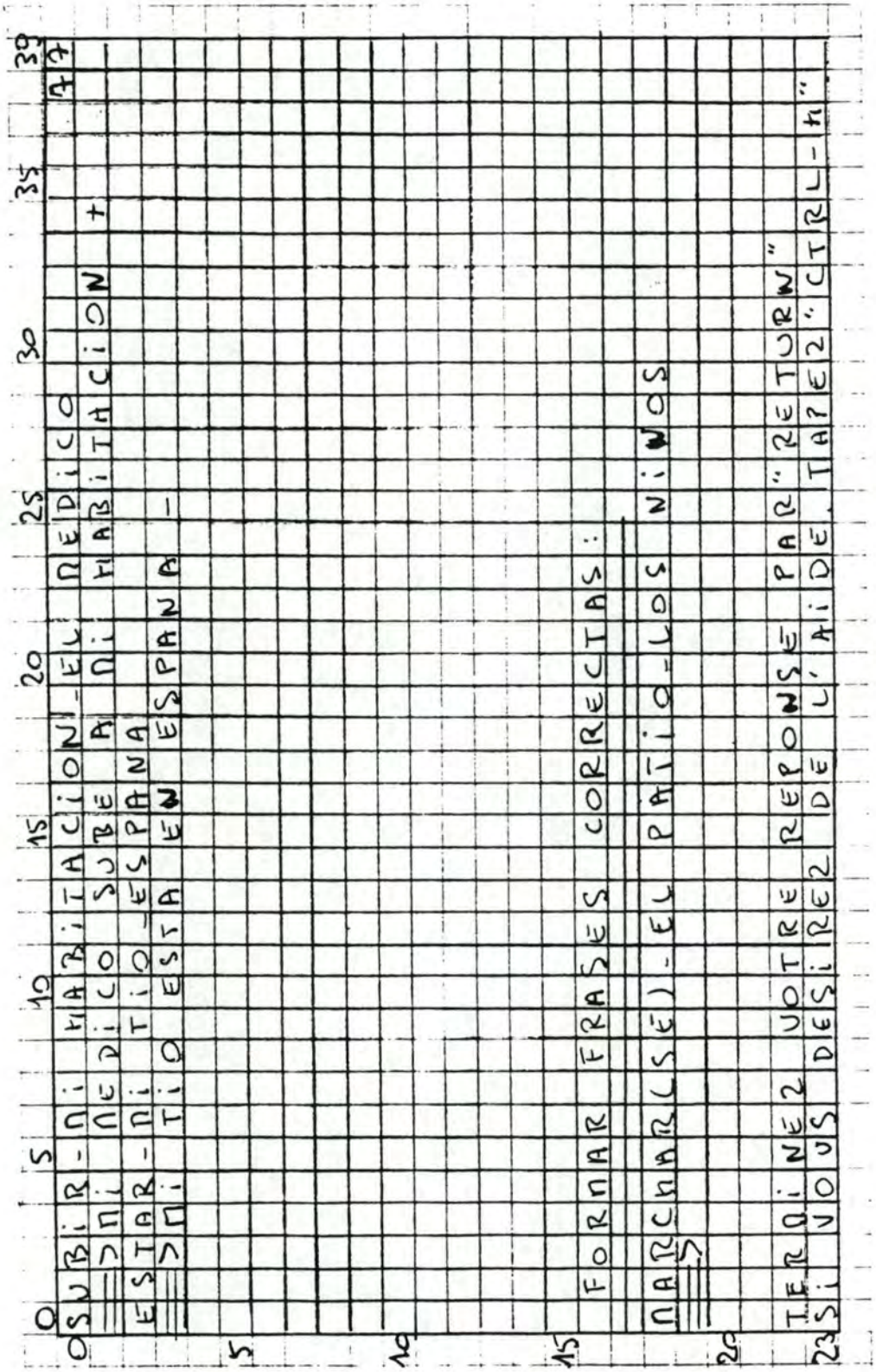
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 18 : introduction au type 4 a.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 4 a.
Sa description est similaire à celle de l'écran 9 : exercices de type 1.

Ecran 20 : Introduction au type 4 b.



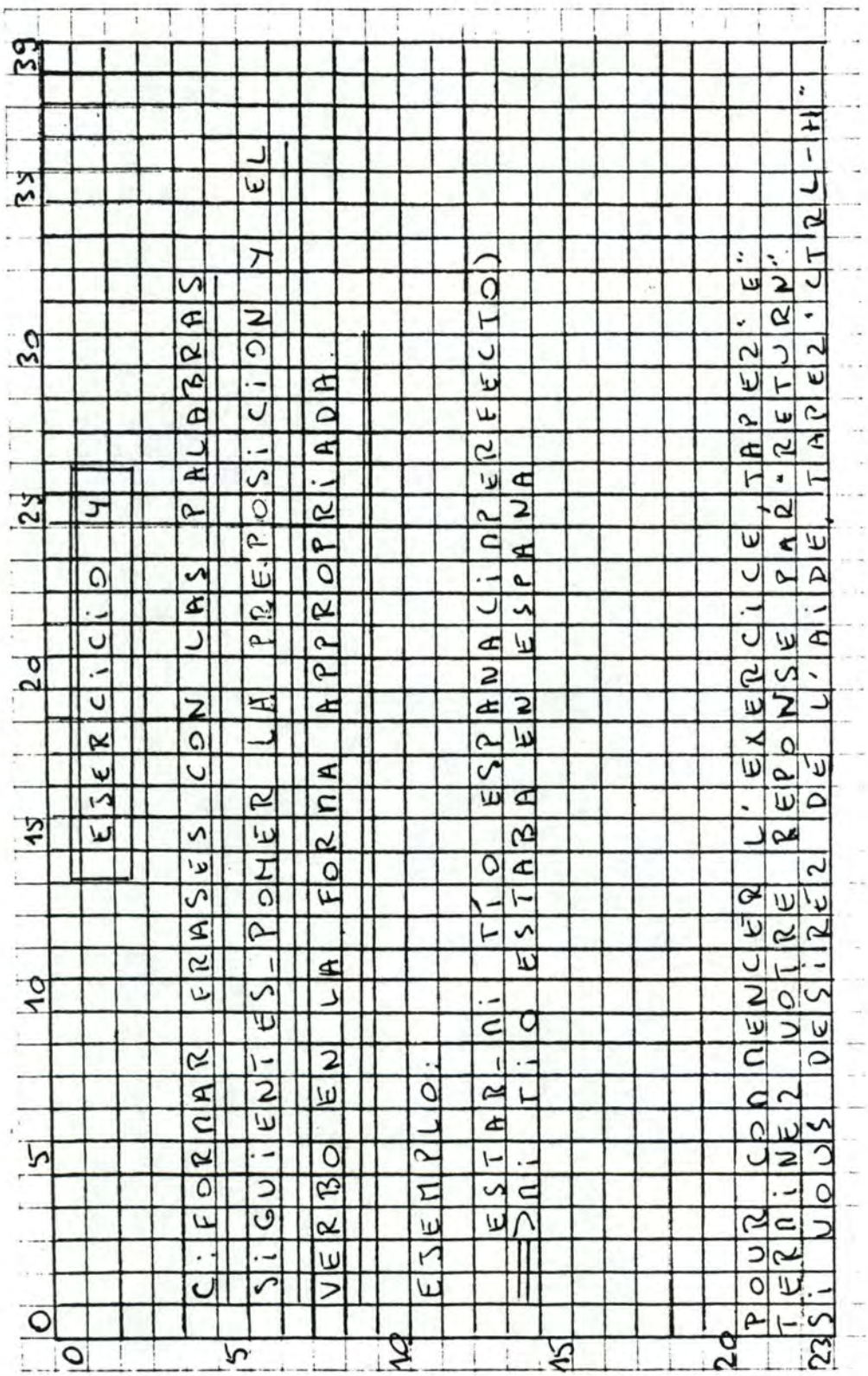
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 2 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 21 : exercices de type 4 b.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 21 : Exercices de type 4 b.



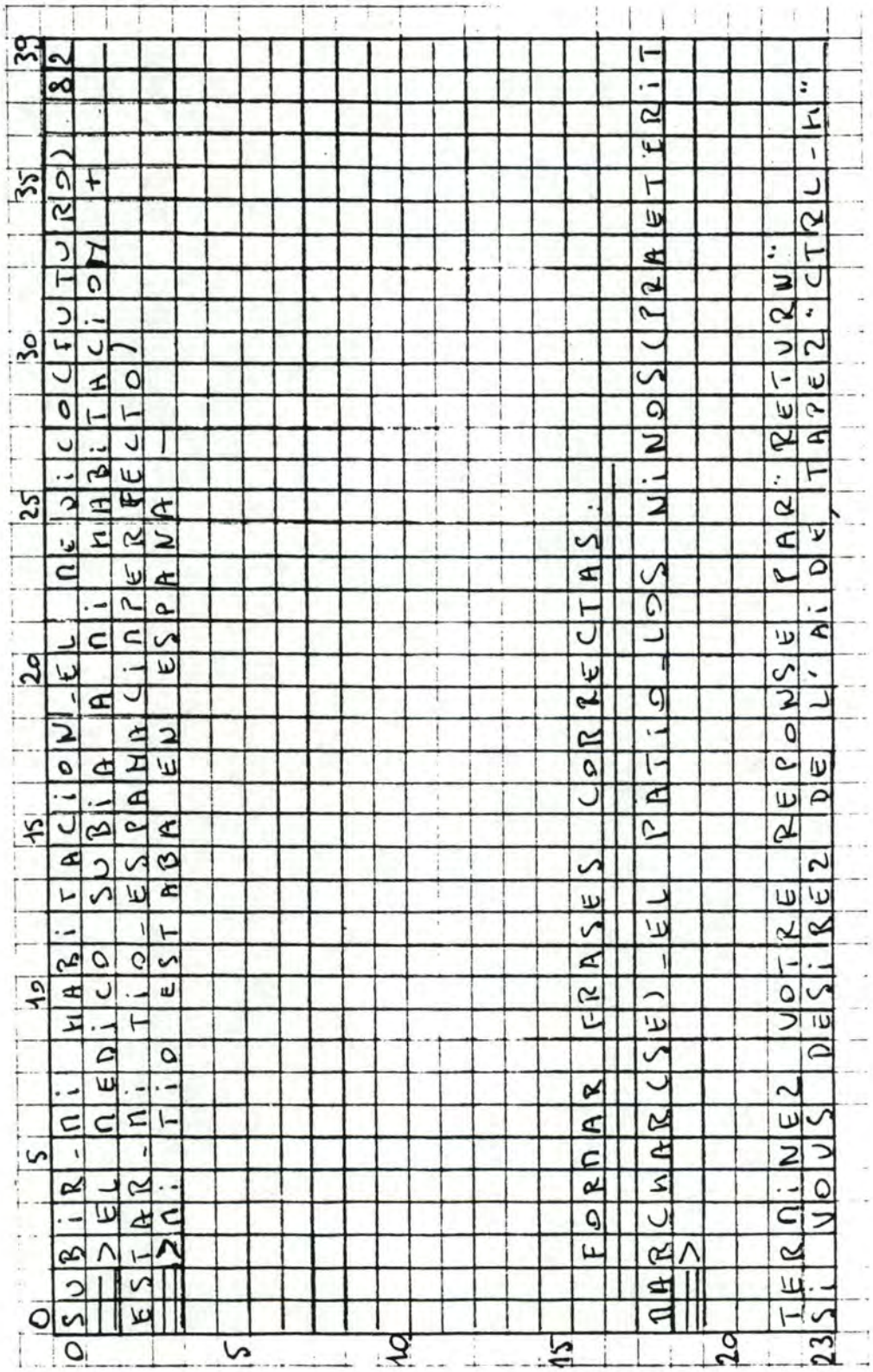
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 20 : introduction au type 4 b.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 4 b.
Sa description est similaire à celle de l'écran 9 : exercices de type 1.

Ecran 22 : Introduction au type 4 c.



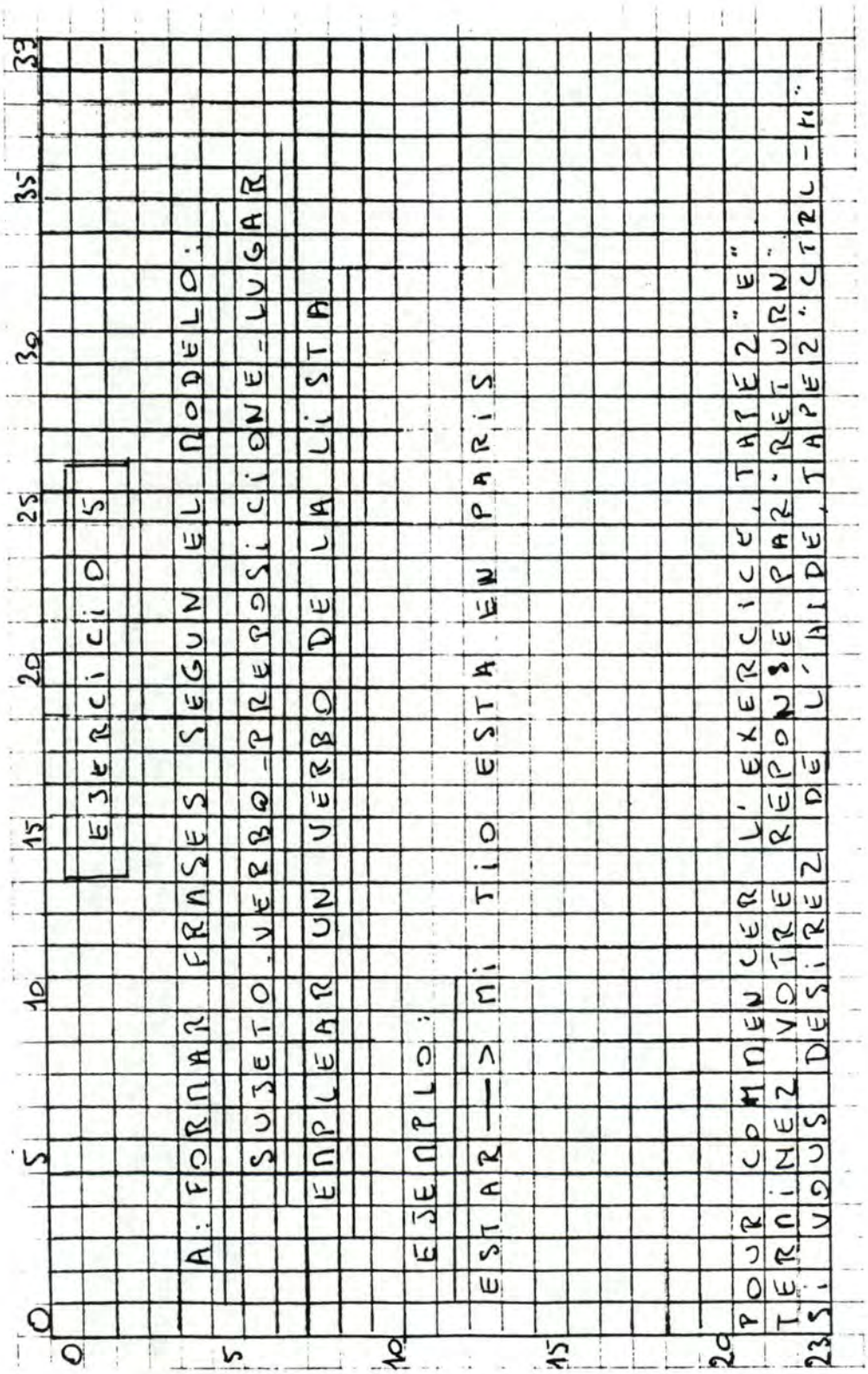
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 2 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 23 : exercices de type 4 c.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 23 : Exercices de type 4 c.



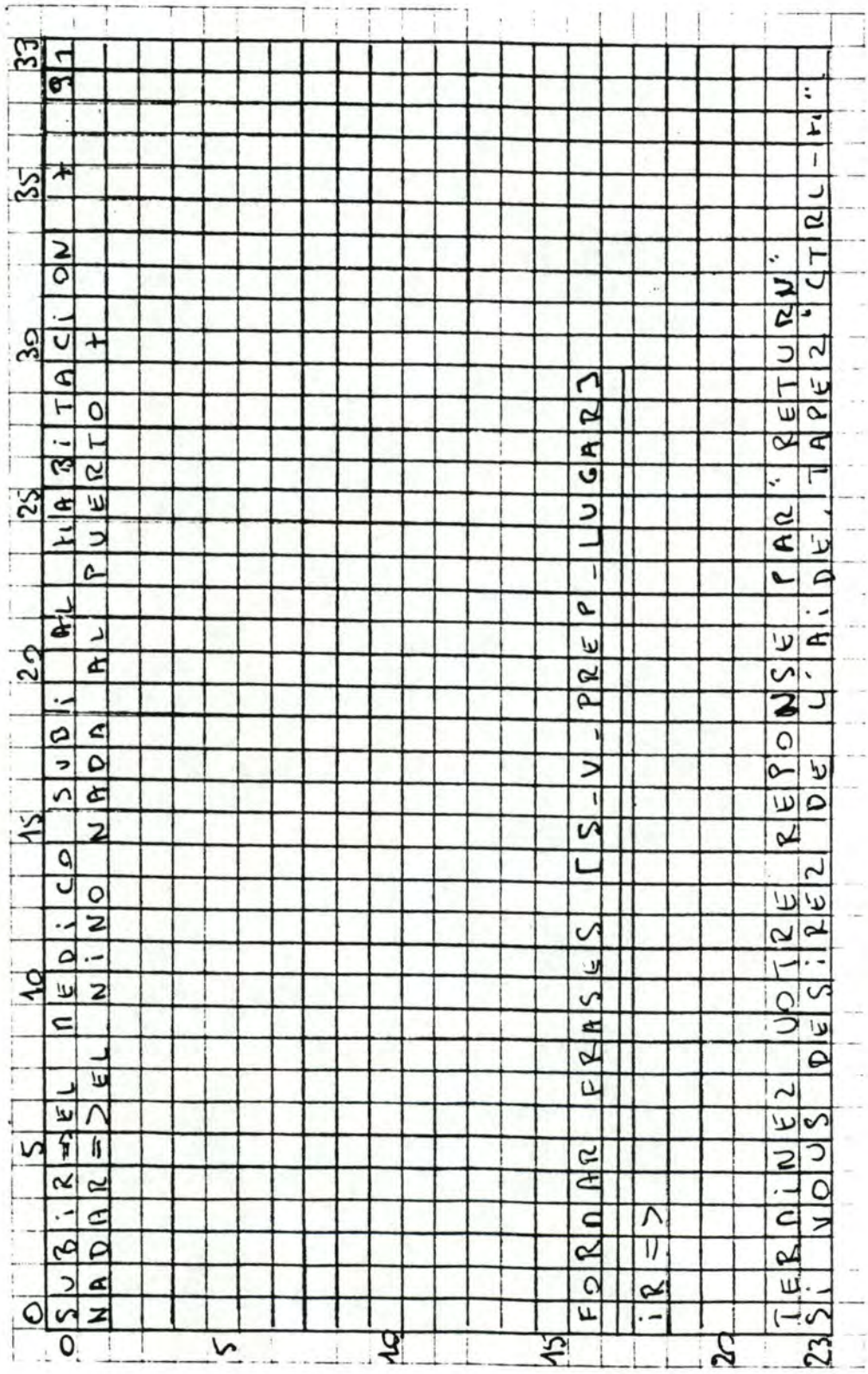
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 22 : introduction au type 4 c.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 4 c.
Sa description est similaire à celle de l'écran 9 : exercices de type 1.

Ecran 24 : Introduction au type 5 a.



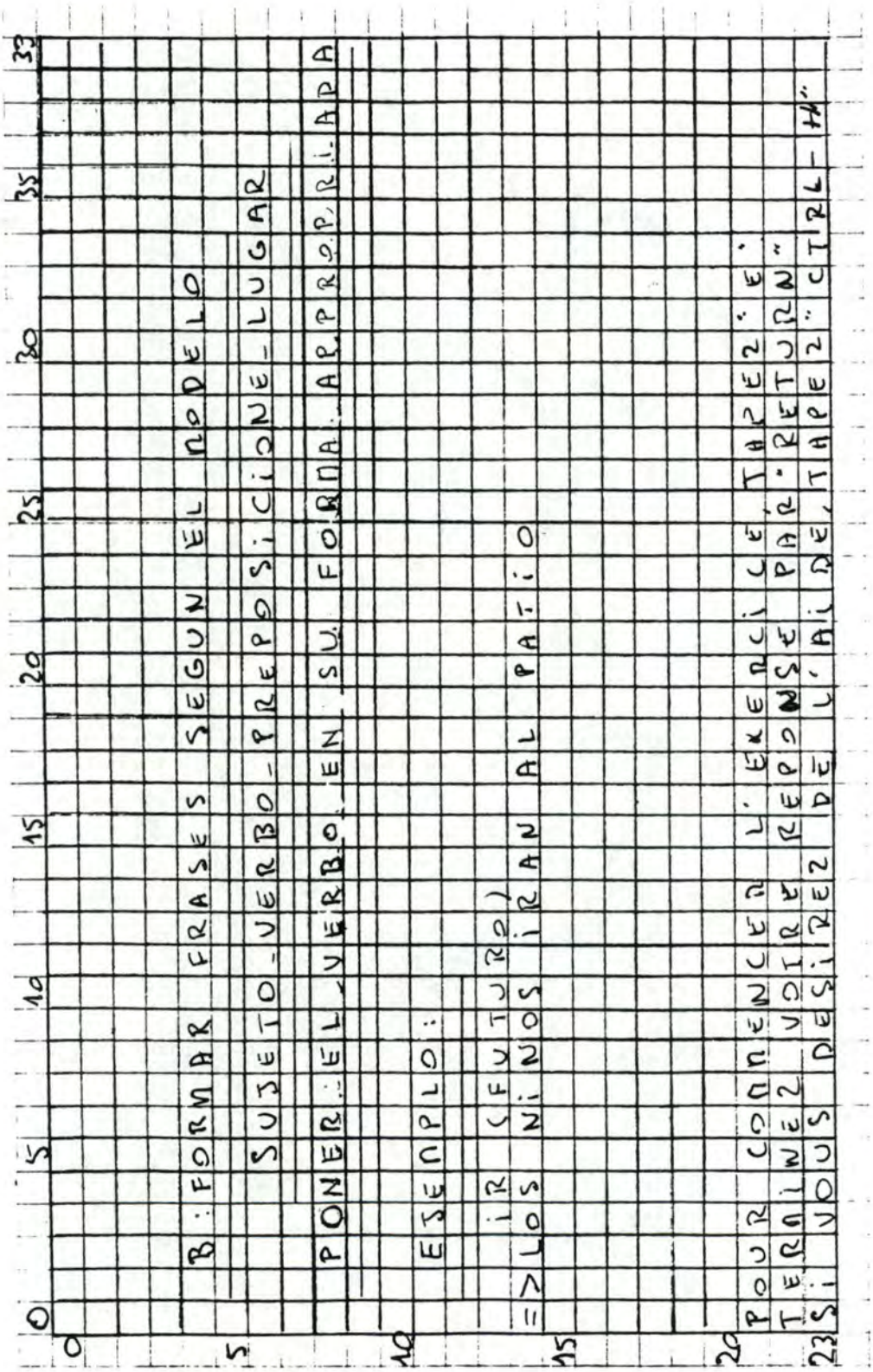
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 2 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 25 : exercices de type 5 a.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 25 : Exercices de type 5 a.



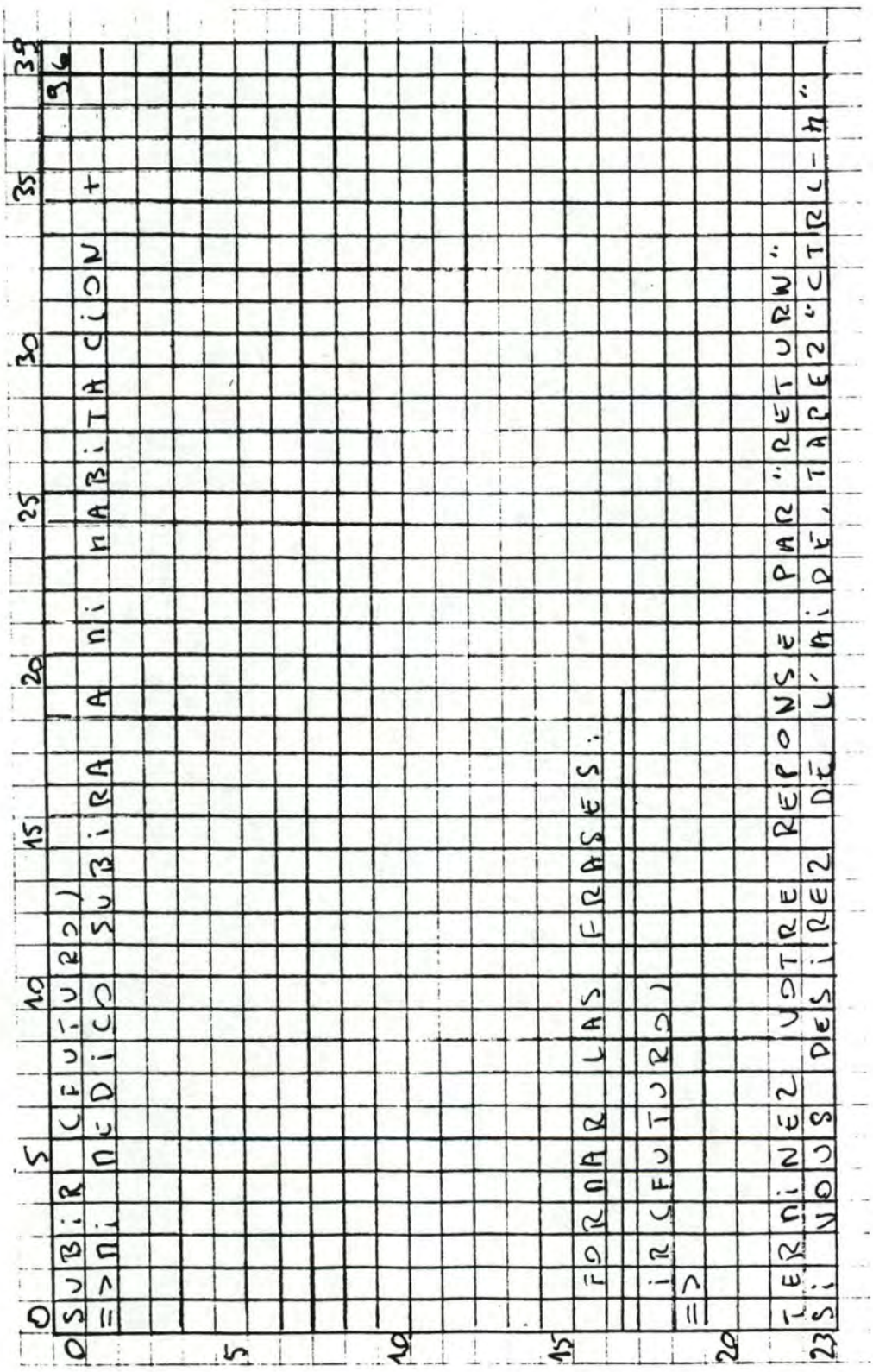
L'utilisateur accède à cet écran par l'intermédiaire de l'écran 24 : introduction au type 5 a.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 5 a.
Sa description est similaire à celle de l'écran 9 : exercices de type 1.

Ecran 26 : Introduction au type 5 b.



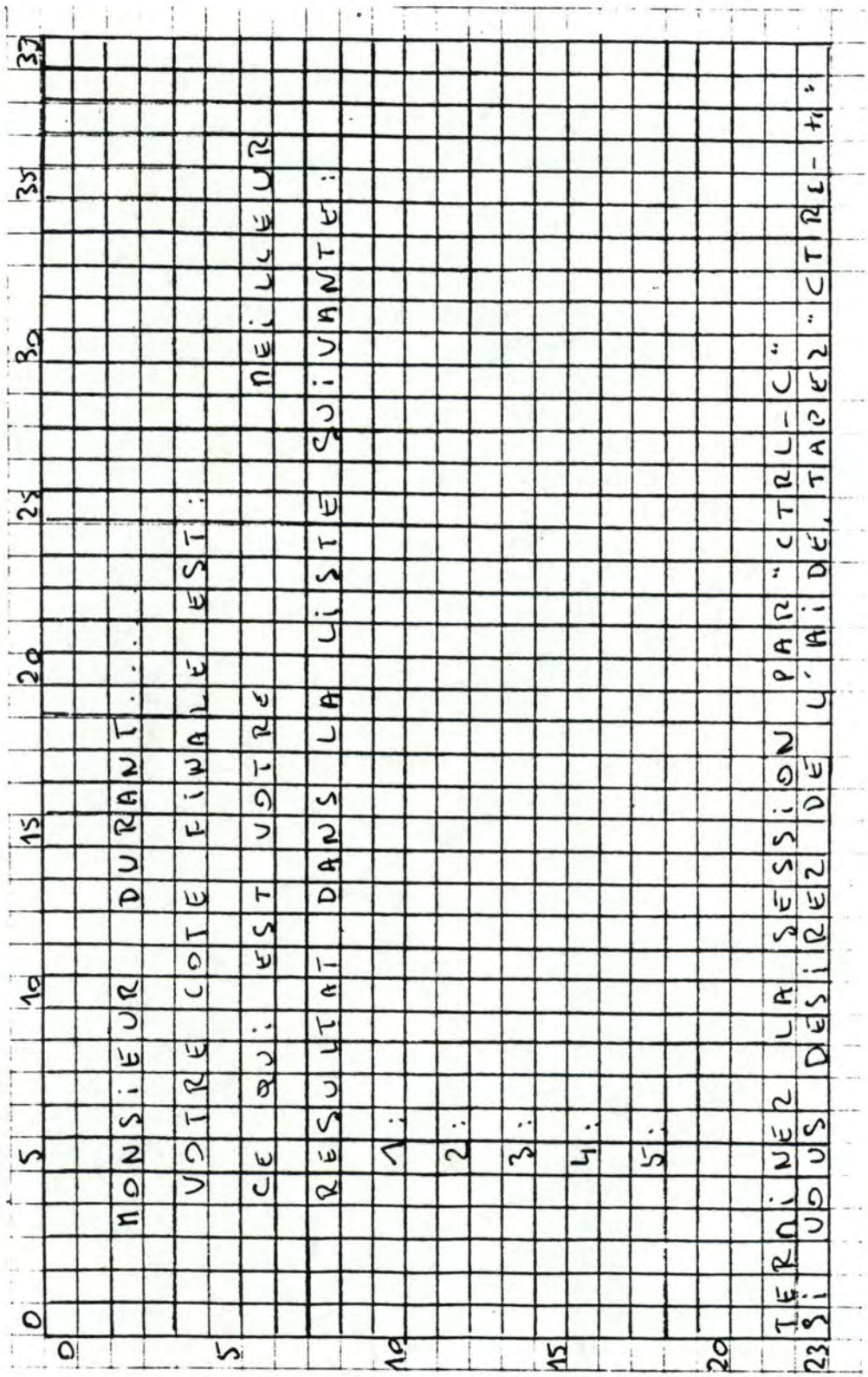
Si l'utilisateur a déjà accompli un type d'exercices durant la présente session, il accédera à cet écran par l'intermédiaire de l'écran des exercices du type précédent.
Dans le cas contraire, il y accédera en venant de l'écran 2 : introduction au système.
Il peut également y revenir après avoir demandé de l'aide.
Il y prend connaissance de l'intitulé de l'exercice ainsi que d'un exemple de résolution de celui-ci.
Lorsque l'utilisateur tape "E", il accède à l'écran 27 : exercices de type 5 b.
S'il tape "CTRL-H", il accède à l'écran 3 : menu d'aide.

Ecran 27 : Exercices de type 5 b.



L'utilisateur accède à cet écran par l'intermédiaire de l'écran 26 : introduction au type 5 b.
Il peut également y revenir après avoir demandé de l'aide.
Cet écran permet à l'utilisateur d'accomplir les exercices de type 5 b.
Sa description est similaire à celle de l'écran 9 : exercices de type 1.

Ecran 28 : Sortie du système.



L'utilisateur accède à cet écran par l'intermédiaire de l'écran des exercices du dernier type de sa session après avoir résolu tous les exercices de ce type qui lui ont été soumis.
Il permet à l'utilisateur de prendre connaissance de sa cote finale ainsi que de la valeur relative de celle-ci par rapport à d'autres cotes réalisées par d'autres élèves et/ou par lui-même.

L'utilisateur accédera à cet écran lorsqu'il aura donné une réponse incorrecte à un des exercices.
Il lui permet de dialoguer avec le système pour corriger sa réponse.
Une fois celle-ci correcte, l'écran 29 disparaît pour faire place à l'écran d'exercices d'où l'utilisateur était parti.

A N N E X E 2.

LISTE DES SUBSTANTIFS DU DICTIONNAIRE EMPLOYE PAR LE MODULE DE CORRECTION D'ORTHOGRAPHE.

AGUA - AIRE - ALMACEN - AMIGO - ANO - APARADOR - APENDICITIS - ARBOL - ARMARIO - ARTISTA - ASESINO - ASESINATO - ATAQUE - AUTOBUS - AVENIDA - AZULEJO - BALCON - BARRIO - BENEFICIENCIA - BOTON - BRASERO - BRAZO - BROCHA - BURRO - CAFETERA - CAJA - CALLE - CALOR - CAMA - CAMINO - CAMISA - CAMPO - CANTIDAD - CARA - CARBON - CARCEL - CARNE - CARTA - CARTEL - CASA - CENA - CERVEZA - CHURRO - CIEGO - CINE - CINTURON - CITA - CIUDAD - COBRE - COCHE - COCIDO - COCINA - COL - COLA - COLOR - COMEDOR - COMISARIO - COMPANIA - CORAZON - COSA - COSTA - CUARTO - CUCHILLO - CUERDA - CUERO - CURSO - DANO - DECIMO - DENTISTA - DESPERTADOR - DESVAN - DIA - DIENTE - DINERO - DIRECCION - DISCO - DISCUSION - DOCTOR - DUENO - EDIFICIO - EFECTO - EJERCITO - EMPLEADO - ENFERMEDAD - ENSALADA - ESCALOYA - ESCOPETA - ESPADA - ESPALDA - ESPOSA - ESPOSO - ESQUINA - ESTACION - ESTATUA - ESTILO - EXITO - FABRICA - FAENA - FALDA - FAMIMIA - FIEBRE - FLOR - FORMA - FOTO - FRESA - FRESON - FREO - FRUTA - FUENTE - FUTBOL - GAFAS - GARAGE - GARBANZO - GATO - GENTE - GITANO - GUITARRA - HABITACION - HALLAZGO - HOMBRE - HOMBRO - HORA - HOSPITAL - HUESPED - HUEVO - IGLESIA - INSTRUMENTO - INVIERNO - JALEO - JAMON - JARRO - JEFE - JOTA - JUDIA - KILO - KILOMETRO - LABOR - LANA - LASTIMA - LENA - LIBRO - LLUVIA - LOCO -LUGAR - LUZ - MACETA - MADEJA - MALETA - MANO - MANTILLA - MANZANA - MANANA - MAPA - MAQUINA - MAR - MARAVILLA - MECANICO - MEDICINA - MEDICO - MERCADO - MERLUZA - MES - MESA - MESON - METRO - MINUTO - MOLINO - MONTANA - MONTE - MUDO - MUJER - MUSICA - NARANJA - NARIZ - NEGRO - NIETO - NINO - NOCHE - NOMBRE - NOVIO - NUMERO - OBRA - OCASION - OJO - OREJA - ORQUESTA - PADRE - PAELLA - PAGINA -PAISAJE - PAJARO - PALACIO - PAN - PAR - PARAGUERO - PARED - PARQUE - PARTE - PARTIDO - PASEO - PATATA - PATIO - PEINETA - PELICULA - PELO - PENSION - PERIODICO - PERRO - PESADILLA - PESCA - PESCADO - PESETA - PEZ - PIE - PIEDRA - PIEZA - PINTOR -

PISO - PLAYA - PLAZA - POLICIA - PORTAL - PORTERA - PRACTICAS - PRECIO - PRIMO - PRISA - PROBLEMA - PROFESION - PROVINCIA - PUEBLO - PUENTE - PUERTA - PUERTO - PULSERA - PURO - RATO - RATON - RXCIBIDOR - RECIPIENTE - REGION - RELOJ - RESTAURANTE - RICO - RINCON - RIO - ROMANCE - ROPA - RUBIA - RUIDO - SALA - SALUD - SANGRIA - SANTA - SANTO - SED - SELLO - SEMANA - SENOR - SERENO - SIERRA - SILLA - SITIO - SITUACION - SOBRINA - SOBRINO - SOL - SOMBRERO - SOTANO - SUELO - SUENO - SUERTE - SUSTO - TACON - TALLER - TAPA - TAPETE - TARDE - TASCA - TEATRO - TELEVISION - TEMPORADA - TERRAZA - TIEMPO - TIO - TIRO - TOCADISCOS - TOMATE - TOMBOLA - TONTA - TONTO - TORERO - TORO - TORTILLA - TRABAJO - TRAJE - TREN - VACA - VACACION - VECINO - VENTANA - VERANO - VERDAD - VERDURA - VEZ - VIAJE - VIEJO - VINO - ZAPATO - ZARZUELA - ZONA - ZUMO.

```
(*  ANNEXE 3:PROGRAMME DE CORRECTION AUTOMATIQUE D'ORTHOGRAPHE PAR LA DISTANCE.
 ==========================================================================*)

PROGRAM CORDIST ;
(*===============*)

(*FONCTION:* LIT "ENTREE",MOT INTRODUIT PAR L'UTILISATEUR
          * VERIFIE S'IL SE TROUVE DANS LE DICTIONNAIRE
             - SI OUI:SON ORTHOGRAPHE EST CORRECTE
             - SI NON:CALCULE LA DISTANCE LE SEPARANT DE CHAQUE MOT DU D
                      DICTIONNAIRE ET PROPOSE LE MOT LE PLUS PROCHE COMME
                      CORRECTION.
          * DEMANDE S'IL FAUT CONTINUER.
             - SI OUI:DEMANDE UN AUTRE MOT ET RECOMMENCE TOUT LE TRAITEMENT
             - SI NON:LE PROGRAMME EST TERMINE.*)

TYPE FICHIER = FILE OF STRING;
     TAB315=ARRAY[1..315] OF STRING[20];
VAR REPUTIL:CHAR;
    IND,I,DIST,DISTMIN:INTEGER;
    INTER,ENTREE,MEILLEUR:STRING;
    ORTHO:BOOLEAN;
    DICOSUBST:FICHIER;
    LISTMOT:TAB315;

PROCEDURE CHARGEMENT;

(*FONCTION:CHARGE LE FICHIER "DICOSUBST"('#5:DICTION.DATA') DANS LE TABLEAU DE
           STRING "LISTMOT"*)

VAR IND:INTEGER;
BEGIN
  RESET(DICOSUBST,'D1:DICTION.DATA');
  FOR IND:=1 TO 315 DO
    BEGIN
       LISTMOT[IND]:=DICOSUBST^;
       GET(DICOSUBST)
    END;
  CLOSE(DICOSUBST,LOCK);
END;

PROCEDURE RECHERCHE (ENTREE:STRING;VAR ORTHO:BOOLEAN);

(*FONCTION:MET "ORTHO" A VRAI SI LE MOT "ENTREE" EST DANS "LISTMOT" ET A
           FAUX DANS LE CAS CONTRAIRE.*)

VAR BI,BS,DEMI:INTEGER;
    CONTINUE:BOOLEAN;
BEGIN
  ORTHO:=FALSE;CONTINUE:=TRUE;BI:=1;BS:=315;
  WHILE (CONTINUE)AND(BI<=BS) DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      IF ENTREE=LISTMOT[DEMI]
            THEN
             BEGIN
              CONTINUE:=FALSE;
              ORTHO:=TRUE
             END
            ELSE IF ENTREE<LISTMOT[DEMI]
                  THEN BS:=DEMI-1
                  ELSE BI:=DEMI+1
    END
END;

PROCEDURE DISTANCE(ENTREE:STRING;STR:STRING;VAR DIST:INTEGER);
```

```
(*ARGUMENTS:- ENTREE:PREMIER MOT
             - STR:SECOND MOT

  RESULTAT:DIST:DISTANCE ENTRE "ENTREE" ET "STR".

  FONCTION:CALCULE LA DISTANCE "DIST" EXISTANT ENTRE "ENTREE" ET "STR".*)

  VAR I,J,LONG:INTEGER;
    STRBIS:STRING;
    RESULTAT:BOOLEAN;

PROCEDURE INTERVERTIR(VAR RESULTAT:BOOLEAN);

(*RESULTAT:RESULTAT:BOOLEEN

  FONCTION:MET "RESULTAT" A VRAI SI LA "INDIEME" LETTRE DE "ENTREE" EST IDENTI-
           TIQUE A LA "(IND2+1IEME)" LETTRE DE "STRBIS" ET SI LA ("IND+1IEME)"
           LETTRE DE "ENTREE" EST IDENTIQUE A LA "IND2IEME" LETTRE DE "STRBIS"
           ET A FAUX DANS LE CAS CONTRAIRE.*)

BEGIN
  RESULTAT:=FALSE;
  IF ((J+1) <= LENGTH(STRBIS)) AND ((I+1) <= LENGTH(ENTREE))
    THEN IF (ENTREE[I]=STRBIS[J+1]) AND (ENTREE[I+1]=STRBIS[J])
           THEN
             BEGIN
               DIST:=DIST+1;
               I:=I+2;
               J:=J+2;
               RESULTAT:=TRUE
             END
END;

PROCEDURE SUBSTITUER(VAR RESULTAT:BOOLEAN);


(*RESULTAT:RESULTAT:BOOLEEN

  FONCTION:MET "RESULTAT" A VRAI SI LA "(IND+1IEME)" LETTRE DE ENTREE EST
           IDENTIQUE A LA "(IND2+1IEME)" LETTRE DE "STRBIS" ET A FAUX SINON.*)

BEGIN
 RESULTAT:=FALSE;
 IF((I+1) <= LENGTH(ENTREE)) AND ((J+1) <= LENGTH(STRBIS))
   THEN IF ENTREE[(I+1)]=STRBIS[(J+1)]
     THEN
       BEGIN
         DIST:=DIST+1;
         I:=I+2;
         J:=J+2;
         RESULTAT:=TRUE
       END
END;

PROCEDURE AJOUTER(VAR RESULTAT:BOOLEAN);

(*RESULTAT:RESULTAT:BOOLEEN

  FONCTION::MET "RESULTAT A VRAI SI LA "(IND+1IEME)" LETTRE DE "ENTREE" EST ID
            IDENTIQUE A LA "(IND2IEME)" LETTRE DE "STRBIS" ET A FAUX SINON.*)

BEGIN
  RESULTAT:=FALSE;
  IF (I+1) <= LENGTH(ENTREE)
    THEN IF ENTREE[I+1]=STRBIS[J]
      THEN
        BEGIN
          DIST:=DIST+1;
```

```
            LONG:=LONG+1;
            I:=I+2;
            J:=J+1;
            RESULTAT:=TRUE
          END
END;

PROCEDURE OUBLIER(VAR RESULTAT:BOOLEAN);

(*RESULTAT:RESULTAT:BOOLEEN

  FONCTION:MET "RESULTAT" A VRAI SI LA "INDIEME" LETTRE DE "ENTREE" EST IDEN-
           TIQUE A LA "(IND2+1IEME)"LETTRE DE "STRBIS" ET A FAUX DANS LE CAS
           CONTRAIRE.*)

BEGIN
  RESULTAT:=FALSE;
  IF (J+1) <= LENGTH(STRBIS)
    THEN IF ENTREE[I]=STRBIS[J+1]
      THEN
        BEGIN
          DIST:=DIST+1;
          LONG:=LONG-1;
          I:=I+1;
          J:=J+2;
          RESULTAT:=TRUE
        END
END;

(* DEBUT PROCEDURE DISTANCE *)

BEGIN
  STRBIS:=STR;
  LONG:=LENGTH(STRBIS);
  DIST:=0;
  I:=1;
  J:=1;
  WHILE (I<=LENGTH(ENTREE)) AND (J<=LENGTH(STRBIS)) DO
    BEGIN
      IF ENTREE[I]=STRBIS[J]
        THEN
          BEGIN
            I:=I+1;
            J:=J+1
          END
        ELSE
          BEGIN
            INTERVERTIR(RESULTAT);
            IF NOT RESULTAT
              THEN
                BEGIN
                  SUBSTITUER(RESULTAT);
                  IF NOT RESULTAT
                    THEN
                      BEGIN
                        AJOUTER(RESULTAT);
                        IF NOT RESULTAT
                          THEN
                            BEGIN
                              OUBLIER(RESULTAT);
                              IF NOT RESULTAT
                                THEN
                                  BEGIN
                                   IF(I=LENGTH(ENTREE))OR(J=LENGTH(STRBIS))
                                       THEN DIST:=DIST+1
                                       ELSE DIST:=DIST+2;
                                   I:=I+1;
                                   J:=J+1
```

```
                                        END
                                    END
                                END
                            END
                        END
    END;
  DIST:=DIST+ABS(LENGTH(ENTREE)-LONG)
END;


(* PROGRAMME PRINCIPAL *)

BEGIN
 CHARGEMENT;
 IND:=0;
 WRITE ('VOULEZ-VOUS COMMENCER ? (Y/N) : ');
 READLN (REPUTIL);
 WHILE REPUTIL <> 'N' DO
  BEGIN
   IND:=IND+1;
   WRITE ('MOT ',IND,' : ');
   READLN(ENTREE);
   RECHERCHE (ENTREE,ORTHO);
   IF ORTHO
    THEN WRITELN('ORTHOGRAPHE CORRECTE')
    ELSE
     BEGIN
       DISTMIN:=MAXINT;
       I:=1;
       WHILE (I<316) AND (DISTMIN>1) DO
         BEGIN
           INTER:=LISTMOT[I];
           DISTANCE(ENTREE,INTER,DIST);
           IF DIST<DISTMIN
              THEN
                BEGIN
                  MEILLEUR:=INTER;
                  DISTMIN:=DIST
                END;
           I:=I+1
         END;
       WRITELN ('PROPOSITION DE CORRECTION:',MEILLEUR)
     END;
     WRITE ('VOULEZ-VOUS CONTINUER ?(O/N)');
     READLN(REPUTIL)
  END
END.
```

A N N E X E 4.

SPECIFICATIONS DE CORMOT (MODULE DE CORRECTION D'ORTHOGRAPHE).

Programme CORMOT.

Arguments:-ENTREE : mot entré au terminal sur proposition du programme.
-DICOSUBST : fichier des mots connus.

Résultat :-dit si "ENTREE" est correctement orthographié.
- sinon propose une correction qui est un mot appartenant au dictionnaire de référence du programme et qui est
  - si "ENTREE" compte plus de 6 lettres, le mot qui en est le plus proche selon la procédure "ESCAPE"
  - si "ENTREE" ne compte pas plus de 6 lettres, un mot obtenu après une modification de "ENTREE" telle que interversion de lettres, suppression de lettre, remplacement de lettre ou insertion de lettre. Si ce programme ne parvient pas à proposer une correction, il répond qu'il ne connaît pas "ENTREE".

Fonction : le programme reçoit des mots un par un au terminal. Pour chaque mot, il vérifie s'il se trouve dans son dictionnaire de référence. Si oui, il répond que l'orthographe est correcte. Si non, il essaie de le corriger de la manière suivante. Si le mot compte plus de 6 lettres, il fait appel à la procédure "ESCAPE" et proposera le mot donné par celle-ci comme solution. Sinon, il modifie le mot jusqu'à ce que le mot ainsi construit soit dans le dictionnaire ou jusqu'à ce qu'il arrive au bout des opérations suivantes en travaillant sur le plus grand nombre de lettres initiales que le mot a en commun avec le dictionnaire : interversion de lettres, suppression de lettre, remplacement de lettre et insertion de lettre.

Procédure AJOUTER.

Résultat : - RESULTAT : booléen.

Fonction : met "RESULTAT" à vrai si la ("IND"+1)^{ème} lettre de "ENTREE" est identique à la ("IND"2)^{ème} lettre de "STRBIS" et à faux sinon.

Procédure CALPRODIGMOT.

Arguments : - ENTREE : mot dont on calcule le produit des digrammes formés par les "COMMAX" premières lettres.
- DIGRAM : matrice ["ↀ" .. " Z", "ↀ".."Z"] d'entier.
- COMMAX : nombre maximum de lettres communes entre "ENTREE" et un mot du dictionnaire.

Résultats : - FREQUENCE : tableau d'entier dont les éléments sont les fréquences des digrammes que forment entre elles les "COMMAX" premières lettres de "ENTREE".
- PROD : tableau d'entier dont les éléments sont les produits des fréquences de digrammes que chacune des "COMMAX" premières lettres forme avec sa précédente et avec sa suivante.
- DIMFREQ : dimension de "FREQUENCE".
- DIMPROD : dimension de "DIMPROD".
- FREQNUL : entier égal à 0 si aucun digramme (donc aucune lettre) n'a de fréquence nulle et égal à la position du premier digramme qui a une fréquence nulle s'il existe.

Fonction : - si aucun digramme de "ENTREE" n'a de fréquence nulle, calcule les tableaux "FREQUENCE" et "PROD" pour les "COMMAX" premières lettres de "ENTREE" et met "FREQNUL" à 0.
- si un digramme a une fréquence nulle, ne se préoccupe pas de "FREQUENCE" et "PROD" mais garnit la variable "FREQNUL" avec la position de ce digramme.

Procédure CHARBYTE.

Arguments : - LISTMOT : tableau des caractères de mots.
- SOURCE : entier, indice dans "LISTMOT" de la première lettre du mot à charger.
- LONGUEUR : entier, longueur en bytes et en nombre de caractères du mot à charger car "LISTMOT" est compacté.

Résultat : - DESTINATION : string où l'on charge le mot commençant à "LISTMOT [SOURCE]" et comptant "LONGUEUR" caractères.

Fonction : charge dans "DESTINATION" le mot dont l'indice dans "LISTMOT" est "SOURCE" et ayant "LONGUEUR" caractères.

Procédure CHARGEMENT.

Arguments : - DICOSUBST ("#5 : diction. data") : fichier des mots connus classés par ordre alphabétique.

Résultats : - LISTMOT : tableau des caractères des éléments (mots) de "DICOSUBST", chaque élément (mot) étant précédé et suivi par le caractère séparateur " ↻ ".

- INDICE : tableau dont les éléments sont les indices des premières lettres de chaque mot dans "LISTMOT".
- INDICETERM : tableau dont les éléments sont les indices de l'antépénultième lettre de chaque mot de plus de 6 lettres du fichier classés par ordre alphabétique sur celle-ci
  - DIMCAR : dimension de "LISTMOT".
  - DIMMOT : dimension de "INDICE".
  - DIMTERM : dimension de "INDICETERM".

Fonction : charge en mémoire centrale dans le tableau de caractères "LISTMOT" les articles (strings) du fichier "DICOSUBST," chaque lettre d'un article devenant un élément du tableau et chaque article du fichier étant précédé et suivi d'un caractère séparateur " ↻ " dans "LISTMOT", le dernier mot de "LISTMOT" est "ZZ" pour des raisons de performance de la recherche dichotomique et le tableau "INDICE" reprenant l'indice de la première lettre de chaque mot dans "LISTMOT". Il constitue également le tableau "CONSINDICETERM" reprenant les indices de l'antépénultième lettre de chaque mot de plus de 6 lettres dans "LISTMOT" classés par ordre alphabétique sur celle-ci.

Procédure CLASLETREMP.

Arguments : - DIGRAM : matrice ("ə".."Z", "ə".."Z") d'entier
- LETAV : caractère, indice de ligne de la matrice "DIGRAM".
- LETAP : caractère, indice de colonne de la matrice "DIGRAM".

Résultats : - LETREMP : tableau reprenant les lettres pouvant se trouver entre "LETAV" et "LETAP" dans l'ordre de leur fréquence relative décroissante.
- COMPTLET : dimension de "LETREMP".

Fonction : calcule le produit des fréquences relatives pour les lettres se trouvant entre "LETAV" et "LETAP" dans le dictionnaire et les ordonne dans "LETREMP", qui compte "COMPTLET" éléments, par ordre de fréquence relative décroissante.

Procédure CLASPOSINS.

Arguments : - FREQNUL : entier égal à 0 si aucun digramme (donc aucune lettre) n'a de fréquence nulle, sinon égal à la position du premier digramme ayant une fréquence relative nulle.
- DIMFREQ : dimension de "FREQUENCE".
- FREQUENCE : tableau des fréquences relatives de (COMMAX) premiers digrammes de "ENTREE".

Résultats : - POSDIGCRIT : tableau d'entier reprenant les positions des digrammes classés en ordre de fréquence relative croissante.
- DIMPOSDIGCRIT : dimension de "POSDIGCRIT".

Fonction : garnit le tableau "POSDIGCRIT" des positions des éléments du tableau "FREQUENCE" classés en ordre de valeur croissante.

Procédure COMPARE.

Arguments : - ENTREE : string, 1° mot.
- FICHMOT : string, 2° mot.

Résultats : - COMC@UR : nombre de lettres initiales communes entre "ENTREE" et "FICHMOT".

Fonction : calcule le nombre de lettres initiales communes entre "ENTREE" et "FICHMOT".

Procédure CONSMATDIG.

Arguments : - LISTMOT : tableau de caractères des mots du dictionnaire.
- DIMCAR : dimension de "LISTMOT".

Résultat : - DIGRAM : matrice ["ↂ" .. "Z", "ↂ" .. "Z"] d'entier..

Fonction : - constitue la matrice "DIGRAM" de fréquence relative des digrammes du dictionnaire. Il calcule le nombre de digrammes se trouvant dans LISTMOT en comptant le nombre de paires différentes de caractères.
Un espace est représenté par "ↂ". Il ne tient pas compte du dernier mot de LISTMOT "ZZ".

Procédure CONTCARCOM.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices des premières lettres de chaque mot dans "LISTMOT".
- ENTREE : string, mot que l'on traite.
- BI : entier, position dans "INDICE" du mot avant lequel devrait se trouver "ENTREE" s'il était dans "LISTMOT" et donc devant lequel il devrait être inséré.
- BS : entier, position dans "INDICE" du mot après lequel devrait se trouver "ENTREE" s'il était dans "LISTMOT" et donc derrière lequel il devrait être inséré.
- DIMMOT : dimension de "INDICE".

Résultats : - COMMAX : nombre de lettres initiales communes que "ENTREE" possède avec un mot de "LISTMOT".
- POSDEPART : position où devrait se trouver "ENTREE" dans "INDICE" s'il était dans "LISTMOT".

Fonction : met dans "COMMAX" le nombre maximum de lettres initiales communes que "ENTREE" possède avec au moins un mot de "LISTMOT" qui est soit celui venant en "BS"^{ème} position dans "INDICE", soit celui venant en "BI"^{ème} position dans "INDICE".

Procédure CORSTRING.

Argument : -

Résultat : CORSTR : "TAB27"
où "TAB27" = packed array ["@".. "Z" of string[1]];

Fonction : construit une table de correspondance de types.
Si "i" est un caractère, "CORSTR[i]" est le même caractère mais dans une variable de type string[1].

Cette astuce est propre au Pascal UCSD qui n'accepte pas l'insertion d'un caractère de type "char" dans un "string".

Procédure DISTANCE.

Arguments : - ENTREE : premier mot.
- STR : deuxième mot.
- DISTMIN : plus petite distance déjà calculée entre "ENTREE" et un autre mot.

Résultats : - DIST : distance entre "ENTREE" et "STR".
- DISTMIN : min (dist, distmin).

Fonction : calcule la distance existant entre "ENTREE" et "STR" et la met dans "DIST". Si "DIST" est plus petit que "DISTMIN", "DISTMIN" devient égal à "DIST".

Procédure ESCAPE.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices dans "LISTMOT" des premiers caractères de chaque mot dans "LISTMOT".
- INDICETERM : tableau des indices de l'antépénultième lettre de chaque mot de plus de 6 lettres de "LISTMOT" classés par ordre alphabétique sur celle-ci.
- DIMMOT : dimension de "INDICE".
- DIMTERM : dimension de "INDICETERM".
- POSDEPART : entier, position où devrait se trouver "ENTREE" s'il existait dans "LISTMOT".
- ENTREE : string, mot sur lequel on travaille.
- COMMAX : nombre de lettres initiales communes maximum avec un mot de "LISTMOT".

Résultats : - ORTHO : booléen, vrai si "ENTREE" modifiée est dans "LISTMOT", faux sinon.
- ENTREE : modifiée si "ORTHO" est vrai, inchangée sinon.

Fonction : Si "COMMAX" $\geq$ 4, travaille sur le préfixe et propose le mot le moins éloigné pour la procédure "DISTANCE" dans tous ceux qui ont le même préfixe.
Ensuite, quelle que soit la valeur de "COMMAX", fait de même pour le(s) mot(s) ayant le même suffixe de trois lettres.

Procédure INSERER.

Arguments : - VAL : valeur à insérer dans le tableau "VALREMP".
- LET : string (1) à insérer dans le tableau "LETREMP".
- VALREMP : tableau d'entier ordonné par ordre décroissant.
- LETREMP : tableau de string (1) .
- MINCOUR : dimension de "VALREMP" et "LETREMP".

Résultats : - VALREMP : tableau d'entier ordonné par ordre décroissant et comprenant maintenant l'élément "VAL".
- LETREMP : tableau de string (1) comprenant maintenant l'élément "LET".
- MINCOUR : dimension de "VALREMP" et "LETREMP" à la sortie de "INSERER", donc augmentée de 1.

Procédure INSERTION.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices des premières lettres de chaque mot dans "LISTMOT".
- DIMMOT : dimension de "INDICE".
- DIGRAM : matrice ("ↀ".."Z", "ↀ".."Z") d'entiers
- FREQUENCE : tableau des fréquences relatives des "COMMAX" premiers digrammes de "ENTREE".
- ENTREE : mot traité.
- DIMFREQ : dimension de "FREQUENCE".
- FREQNUL : entier égal à 0 si aucun digramme (donc aucune lettre) n'a de fréquence nulle et sinon égal à la position dans le mot "ENTREE" du premier digramme dont la fréquence est nulle.

Résultats : - ENTREE : mot traité si "ORTHO" est vrai.
- ORTHO : vrai si "ENTREE" est dans le dictionnaire, faux sinon.

Fonction : classe les "COMMAX" premiers digrammes de "ENTREE" par ordre de fréquence relative croissante puis dans cet ordre insère entre chaque composante de ces digrammes une lettre par ordre de fréquence relative décroissante des lettres jusqu'à ce que le mot ainsi formé soit dans le dictionnaire ou qu'il n'y ait plus de lettres et plus de digrammes.

Procédure INTERCHANGER.

Résultat : - RESULTAT : booléen.

Fonction : met "RESULTAT" à vrai si la "IND"^{ère} lettre de "ENTREE" est identique à la ("IND"2+1)^{ère} lettre de "STRBIS" et si la ("IND"+1)^{ère} lettre de "ENTREE" est identique à la ("IND"2)^{ème} lettre de "STRBIS" et à faux sinon.

Procédure INTERVERTIR.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices dans "LISTMOT" des premiers caractères de chaque mot dans "LISTMOT".
- INDICETERM : tableau des indices de l'antépénultième lettre de chaque mot de plus de 6 lettres de "LISTMOT" classés par ordre alphabétique sur celle-ci.
- DIMMOT : dimension de "INDICE".
- DIMTERM : dimension de "INDICETERM".
- COMMAX : nombre de lettres sur lesquelles on travaille.
- ENTREE : mot que l'on traite.

Résultats : - ENTREE : mot modifié si "ORTHO" a été mis à vrai.
- ORTHO : vrai si "ENTREE" modifiée est dans "LISTMOT", faux sinon.

Fonction : si la longueur de "ENTREE" est au moins égale à 2, intervertit une à une les "COMMAX" premières lettres de "ENTREE" avec leur suivante immédiate jusqu'à ce que le mot ainsi formé soit dans "LISTMOT" ou qu'on intervertit la "COMMAX"^{ème} lettre avec la "COMMAX+1"^{ème} lettre.

Procédure LETAVAP.

Arguments : - ENTREE : mot sur lequel on travaille.
- POSCRIT : position d'une lettre de "ENTREE".

Résultats : - LETAV : lettre de "ENTREE" venant en (POSCRIT-1)ème position dans "ENTREE". Si "POSCRIT" = 1 alors "LETAV" = (espace).
- LETAP : lettre de "ENTREE" venant en (POSCRIT+1)ème position dans "ENTREE". Si "POSCRIT" = longueur de "ENTREE", alors "LETAP" = (espace)

Fonction : recherche dans "ENTREE" les lettres qui sont avant et après celle qui est (POSCRIT)ème position et les met respectivement dans "LETAV" et "LETAP".

Procédure LETGAUDROI.

Arguments : - TRAVAIL : mot sur lequel on travaille.
- POSCRIT : entier, position du digramme concerné dans "TRAVAIL".

Résultats : - LETGAU : lettre de gauche du (POSCRIT)ème digramme de "TRAVAIL".
- LETDROI : lettre de droite du (POSCRIT)ème digramme de "TRAVAIL".

Fonction : recherche dans "TRAVAIL" les composantes du digramme qui vient en (POSCRIT)ème position et les met dans "LETGAU" et "LETDROI".

Procédure LISTCANDIDAT.

Arguments : - ENTREE : mot à corriger.
- COMMAX : nombre de lettres communes initiales entre "ENTREE" et un mot du dictionnaire - critère de sélection pour la liste.
- POSDEPART : indice que devrait avoir la première lettre de "ENTREE" dans "LISTMOT" si celui-ci s'y trouvait.

Résultats : - DISTMIN : distance minimale entre "ENTREE" et le mot le plus proche dans "LISTMOT".
- INTER : mot ayant ses "COMMAX" premières lettres communes avec "ENTREE" et distant de celui-ci de "DISTMIN".

Fonction : recherche dans "LISTMOT" les mots ayant leurs "COMMAX" premières lettres communes avec "ENTREE" et met celui qui est le plus proche de "ENTREE" dans "INTER", la distance entre ces deux mots étant chargée dans "DISTMIN".

Procédure LISTSUFFCANDI.

Arguments : - SUFFIXE : string (3) = suffixe de trois lettres de "ENTREE", critère de sélection de "LISTSUFFCANDI".
- POSDEPART : position de départ dans la recherche de candidat, adresse dans "INDICETERM" d'un string (3) identique à "SUFFIXE".
- DISTMIN : distance minimale entre "ENTREE" et le meilleur des candidats, déjà sélectionnés s'il existe, "MAXINT" sinon.
- ENTREE : mot à corriger.

Résultats : - DISTMIN : distance minimale entre "ENTREE" et le meilleur des candidats.
- INTER : meilleur des candidats distant de "ENTREE" de "DISTMIN".

Fonction : recherche dans "LISTMOT" les mots ayant le même suffixe de trois lettres que "ENTREE" et met celui qui est le moins éloigné de "ENTREE" dans "INTER" si la distance est inférieure à la valeur de "DISTMIN" passée comme argument.

Procédure MOVVALET.

Arguments : - DEB : entier, début de l'intervalle à déplacer dans "VALREMP" et "LETREMP".
- FIN : entier, dimension de "VALREMP" et "LETREMP" diminuée de 1.
- VALREMP : tableau d'entier.
- LETREMP : tableau de string (1).

Résultats : - VALREMP : tableau d'entier dont les éléments "DEB" et "DEB + 1" sont identiques.
- LETREMP : tableau de string (1) dont les éléments "DEB" et "DEB + 1" sont identiques.

Fonction : déplace les éléments de "VALREMP" et "LETREMP" d'une position vers la gauche à partir du "DEB"ème élément.

Procédure OUBLIER.

Résultat : - RESULTAT : booléen.

Fonction : met "RESULTAT" à vrai si la ("IND")^{ème} lettre de "ENTREE"
est identique à la ("IND"2+1)^{ème} lettre de "STRBIS"
et à faux sinon.

Procédure RECHERCHE.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices des premières lettres de chaque mot dans "LISTMOT".
- DIMMOT : dimension de "INDICE".
- ENTREE : string, mot que l'on recherche.

Résultats : - ORTHO : booléen, vrai si "ENTREE" est dans "LISTMOT", faux sinon.
- BI : entier - position dans "INDICE" du mot avant lequel devrait se trouver "ENTREE" s'il était dans "LISTMOT" et donc devant lequel il devrait être inséré.
- BS : entier - position dans "INDICE" du mot après lequel devrait se trouver "ENTREE" s'il était dans "LISTMOT" et donc derrière lequel il devrait être inséré.

Fonction : renvoie "ORTHO" à vrai si "ENTREE" est dans "LISTMOT", et à faux sinon, avec dans "BI", l'indice dans "INDICE" du mot devant se trouver juste avant "ENTREE" et dans "BS" celui du mot devant se trouver après.

Procédure RECHMINPROD.

Arguments : - PROD : tableau d'entiers positifs ou nuls.
- DIMPROD : dimension de "PROD".
- FREQNUL : = 0 si aucun digramme (donc aucune lettre) n'a de fréquence nulle, sinon = la position dans le mot "ENTREE" du premier digramme dont la fréquence est nulle.

Résultats : - POSLETCRIT : tableau d'entier reprenant les positions des éléments de "PROD" classés en ordre croissant.
- DIMPOSLETCRIT : dimension de "POSLETCRIT".

Fonction : si aucun digramme (donc aucune lettre) n'a de fréquence nulle, alors classer les indices des éléments de "PROD" en ordre croissant de valeurs des éléments dans "POSLETCRIT" et mettre "FREQNUL" à 0.
Sinon mettre dans "FREQNUL" l'indice dans "ENTREE" du premier digramme dont la fréquence est nulle.

Procédure REMPLACEMENT.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices dans "LISTMOT" des premières lettres de chaque mot.
- DIMMOT : dimension de "INDICE".
- DIGRAM : matrice ("◌"..."Z","◌"..."Z") d'entiers.
- DIMPOSLETCRIT : entier, nombre des lettres que l'on va remplacer dans "ENTREE".
- POSLETCRIT : tableau d'entier donnant l'ordre de traitement des lettres.
- ENTREE : mot que l'on traite.

Résultats : - ENTREE : mot traité.
- ORTHO : vrai si "ENTREE" est dans le dictionnaire, faux sinon.

Fonction : pour les "DIMPOSLETCRIT" premières lettres de "ENTREE" et tant que "ENTREE" n'est pas dans "DICOSUBST", remplace une lettre par celles qui peuvent la remplacer à cette place dans l'ordre de leur probabilité une par une et vérifie si le mot ainsi formé est dans "DICOSUBST". Dans ce cas, "ORTHO" est vrai, sinon "ORTHO" est faux.

Procédure REMPLACLETTRE.

Arguments : - POSCRIT : entier, position dans "ENTREE" de la lettre à remplacer.
- ENTREE : mot traité.
- CHOISIE : lettre qui remplace la (POSCRIT)ème de "ENTREE".

Résultats : - ENTREE : mot dont on a remplacé une lettre.

Fonction : remplace dans "ENTREE" la lettre venant en (POSCRIT)ème position par la lettre "choisie".

Procédure SUBSTITUER.

Résultat : - RESULTAT : booléen.

Fonction : met "RESULTAT" à vrai si la ("IND"+1)^{ère} lettre de "ENTREE" est identique à la ("IND"2+1)^{ème} lettre de "STRBIS" et à faux sinon.

Procédure SUPPRESSION.

Arguments : - LISTMOT : tableau des caractères des mots.
- INDICE : tableau des indices des premiers caractères de chaque mot dans "LISTMOT".
- DIMMOT : dimension de "INDICE".
- DIMPOSLETCRIT : entier, nombre de lettres que l'on va retirer de "ENTREE".
- POSLETCRIT : tableau d'entier donnant l'ordre de traitement des lettres.
- ENTREE : mot que l'on traite.

Résultats : - ENTREE : mot traité.
- ORTHO : vrai si "ENTREE" est dans le dictionnaire; faux sinon.

Fonction : supprime l'une après l'autre les "DIMPOSLETCRIT" premières lettres de "ENTREE" dans l'ordre donné par les éléments de "POSLETCRIT" jusqu'à les avoir enlevées toutes ou jusqu'à ce que le mot ainsi formé soit dans "DICOSUBST". Si c'est le cas, "ORTHO" est mis à "true" sinon à "false".

```
(*  ANNEXE 5:MODULE DE CORRECTION AUTOMATIQUE D'ORTHOGRAPHE.
    ========================================================*)

(*$S+*)
PROGRAM CORMOT;
(*=============*)
CONST NBREMOT = 320;
      NBRECAR = 2200;
      NBRETERM =100;
TYPE MATRICE=ARRAY ['@'..'Z','@'..'Z'] OF INTEGER;
     FICHIER = FILE OF STRING;
     CHAINE=STRING[20];
     TABCAR= PACKED ARRAY[1..NBRECAR] OF CHAR;
     TAB7= ARRAY [1..7] OF INTEGER;
     TAB27= ARRAY [1..27] OF CHAINE;
     TABIND= ARRAY[1..NBREMOT] OF INTEGER;
     TABTERM= ARRAY[1..NBRETERM]OF INTEGER;
     STR1=STRING[1];
VAR REPUTIL:CHAR;
    BI,BS,DIMMOT,DIMCAR,DIMTERM,IND,COMMAX,POSDEPART,FREQNUL,DIMPROD,DIMFREQ,
      DIMPOSLETCRIT:INTEGER;
    ENTREE:CHAINE;
    ORTHO:BOOLEAN;
    POSLETCRIT,PROD,FREQUENCE:TAB7;
    DIGRAM:MATRICE;
    INDICE:TABIND;
    INDICETERM:TABTERM;
    LISTMOT:TABCAR;
    DICOSUBST:FICHIER;
    CORSTR:PACKED ARRAY['@'..'Z']OF STR1;

PROCEDURE CHARGEMENT(VAR LISTMOT:TABCAR;VAR INDICE:TABIND;VAR INDICETERM:
                     TABTERM;VAR DIMCAR,DIMMOT,DIMTERM:INTEGER);
VAR IND,IND3,IND4,IND5,DEBTERMENT,INDTERMENT:INTEGER;
    TERMENT:STRING[3];
    TABTERM:ARRAY[1..NBREMOT]OF STRING[3];
BEGIN
  WRITELN('DEBUT DE CHARGEMENT,DUREE:20 SECONDES!');
  IND3:=0;
  IND:=1;
  TABTERM[1]:='ZZZ';
  LISTMOT[IND]:='@';
  DIMTERM:=0;
  RESET(DICOSUBST,'#5:DICTION.DATA');
  WHILE NOT EOF(DICOSUBST) DO
     BEGIN
       IND3:=IND3+1;
       IND:=IND+1;
       INDICE[IND3]:=IND;
       ENTREE:=DICOSUBST^;
       MOVELEFT(ENTREE[1],LISTMOT[IND],LENGTH(ENTREE));
       IND:=IND+LENGTH(ENTREE);
       IF LENGTH(ENTREE) >6 THEN
         BEGIN
           DEBTERMENT:=LENGTH(ENTREE)-2;
           TERMENT:=COPY(ENTREE,DEBTERMENT,3);
           INDTERMENT:=IND-3;
           IND4:=1;
           WHILE ((IND4<=DIMTERM)AND(TERMENT>=TABTERM[IND4])) DO IND4:=IND4+1;
           DIMTERM:=DIMTERM+1;
           IF DIMTERM>1 THEN FOR IND5:=DIMTERM DOWNTO IND4+1 DO
             BEGIN
               TABTERM[IND5]:=TABTERM[IND5-1];
               INDICETERM[IND5]:=INDICETERM[IND5-1]
             END;
           TABTERM[IND4]:=TERMENT;
           INDICETERM[IND4]:=INDTERMENT
         END;
```

```
        LISTMOT[IND]:='@';
        GET(DICOSUBST)
      END;
  INDICE[IND3+1]:=IND+1;
  LISTMOT[IND+1]:='Z';
  LISTMOT[IND+2]:='Z';
  LISTMOT[IND+3]:='@';
  DIMCAR:=IND+3;
  DIMMOT:=IND3+1;
  WRITELN('DIMENSION DE LISTMOT:',DIMCAR);
  WRITELN('NOMBRE DE MOTS:',DIMMOT);
  CLOSE(DICOSUBST,LOCK)
END;

PROCEDURE CORSTRING;
VAR I:CHAR;
BEGIN
    CORSTR['@']:='@';
    CORSTR['A']:='A';
    CORSTR['B']:='B';
    CORSTR['C']:='C';
    CORSTR['D']:='D';
    CORSTR['E']:='E';
    CORSTR['F']:='F';
    CORSTR['G']:='G';
    CORSTR['H']:='H';
    CORSTR['I']:='I';
    CORSTR['J']:='J';
    CORSTR['K']:='K';
    CORSTR['L']:='L';
    CORSTR['M']:='M';
    CORSTR['N']:='N';
    CORSTR['O']:='O';
    CORSTR['P']:='P';
    CORSTR['Q']:='Q';
    CORSTR['R']:='R';
    CORSTR['S']:='S';
    CORSTR['T']:='T';
    CORSTR['U']:='U';
    CORSTR['V']:='V';
    CORSTR['W']:='W';
    CORSTR['X']:='X';
    CORSTR['Y']:='Y';
    CORSTR['Z']:='Z';
END;

PROCEDURE CONSMATDIG(LISTMOT:TABCAR;DIMCAR:INTEGER;VAR DIGRAM:MATRICE);
VAR LIGNE,COLONNE:CHAR;
    I:INTEGER;
BEGIN
  WRITELN('CHARGEMENT DE LA MATRICE.');
  FOR LIGNE:= '@' TO 'Z' DO
    FOR COLONNE:= '@' TO 'Z' DO
      DIGRAM[LIGNE,COLONNE]:=0;
  FOR I:=1 TO DIMCAR-4 DO
    DIGRAM[LISTMOT[I],LISTMOT[I+1]]:=DIGRAM[LISTMOT[I],LISTMOT[I+1]] + 1
END;

PROCEDURE CHARBYTE(LISTMOT:TABCAR;SOURCE:INTEGER;VAR LONGUEUR:INTEGER;
                   VAR DESTINATION:CHAINE);
BEGIN
  LONGUEUR:=LONGUEUR+1;
  MOVELEFT(LISTMOT[SOURCE-1],DESTINATION,LONGUEUR);
  LONGUEUR:=LONGUEUR-1;
  MOVELEFT(LONGUEUR,DESTINATION,1)
END;
```

```
PROCEDURE COMPARE(ENTREE,FICHMOT:CHAINE;VAR COMCOUR:INTEGER);
VAR TROUVE:BOOLEAN;
    IND:INTEGER;
BEGIN
  TROUVE:=TRUE;IND:=1;
  WHILE((IND<=LENGTH(ENTREE))AND(IND<=(LENGTH(FICHMOT)))AND TROUVE)
    DO IF ENTREE[IND] = FICHMOT[IND]
      THEN IND:=IND+1
      ELSE TROUVE:=FALSE;
  COMCOUR:=IND-1
END;

PROCEDURE RECHERCHE (LISTMOT:TABCAR;INDICE:TABIND;DIMMOT:INTEGER;ENTREE:CHAINE;
                     VAR ORTHO:BOOLEAN;VAR BI,BS:INTEGER);
VAR DEMI,INDCAR,LONGUEUR:INTEGER;
    INTER:CHAINE;
    CONTINUE:BOOLEAN;
BEGIN
  ORTHO:=FALSE;
  CONTINUE:=TRUE;
  BI:=1;
  BS:=DIMMOT-1;
  WHILE (CONTINUE)AND(BI<=BS) DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICE[DEMI];
      LONGUEUR:=INDICE[DEMI+1]-INDCAR-1;
      CHARBYTE(LISTMOT,INDCAR,LONGUEUR,INTER);
      IF ENTREE=INTER
            THEN
             BEGIN
              CONTINUE:=FALSE;
              ORTHO:=TRUE
             END
            ELSE IF ENTREE<INTER
                  THEN BS:=DEMI-1
                  ELSE BI:=DEMI+1
    END
 END;(*RECHERCHE*)

 PROCEDURE CONTCARCOM(LISTMOT:TABCAR;INDICE:TABIND;BI,BS,DIMMOT:INTEGER;
                      ENTREE:CHAINE;VAR COMMAX,POSDEPART:INTEGER);
 VAR COMCOUR:INTEGER;

   PROCEDURE COMPMOT(NO:INTEGER;ENTREE:CHAINE;VAR COMMUNE:INTEGER);
   VAR MOT:CHAINE;
       LONGUEUR:INTEGER;
   BEGIN
      LONGUEUR:=INDICE[NO+1]-INDICE[NO]-1;
      CHARBYTE(LISTMOT,INDICE[NO],LONGUEUR,MOT);
      COMPARE(ENTREE,MOT,COMMUNE)
   END;

 BEGIN
  COMCOUR:=0;
  COMMAX:=0;
  IF BS>0 THEN COMPMOT(BS,ENTREE,COMCOUR);
  IF BI<(DIMMOT-1)THEN
                    BEGIN
                      COMPMOT(BI,ENTREE,COMMAX);
                      POSDEPART:=BI
                    END;
  IF COMCOUR>COMMAX
     THEN
        BEGIN
           COMMAX:=COMCOUR;
           POSDEPART:=BS
        END
     ELSE POSDEPART:=BI
END;(*CONTCARCOM*)
```

```
PROCEDURE CALPRODIGMOT(ENTREE:CHAINE;DIGRAM:MATRICE;COMMAX:INTEGER;
            VAR FREQUENCE,PROD:TAB7;VAR DIMFREQ,DIMPROD,FREQNUL:INTEGER);
VAR PREMIND,DEUXIND:CHAR;
    TRAVAIL:CHAINE;
    CONTINUE:BOOLEAN;
    LGTRAV,IND:INTEGER;
BEGIN
  IF(LENGTH(ENTREE)=COMMAX) OR (LENGTH(ENTREE)=COMMAX+1)
    THEN
      BEGIN
        TRAVAIL:=CONCAT('@',ENTREE,'@');
        LGTRAV:=LENGTH(TRAVAIL)
      END
    ELSE
      BEGIN
        TRAVAIL:=CONCAT('@',ENTREE);
        LGTRAV:=COMMAX+3
      END;
  CONTINUE:=TRUE;
  FREQNUL:=0;
  IND:=1;
  DIMPROD:=0;
  DIMFREQ:=0;
  WHILE (IND<LGTRAV) AND CONTINUE DO
    BEGIN
      PREMIND:=TRAVAIL[IND];
      DEUXIND:=TRAVAIL[IND+1];
      DIMFREQ:=DIMFREQ+1;
      FREQUENCE[DIMFREQ]:=DIGRAM[PREMIND,DEUXIND];
      IF DIMFREQ>1
        THEN
          BEGIN
            DIMPROD:=DIMPROD+1;
            PROD[DIMPROD]:=FREQUENCE[DIMFREQ-1]*FREQUENCE[DIMFREQ]
          END;
      IF FREQUENCE[DIMFREQ]=0
        THEN
          BEGIN
            IF IND <> LGTRAV-1
              THEN
                BEGIN
                  DIMPROD:=DIMPROD+1;
                  PROD[DIMPROD]:=0
                END;
            FREQNUL:=IND;
            CONTINUE:=FALSE
          END;
      IND:=IND + 1;
    END
END;

PROCEDURE RECHMINPROD(PROD:TAB7;DIMPROD,FREQNUL:INTEGER;
                      VAR DIMPOSLETCRIT:INTEGER;VAR POSLETCRIT:TAB7);

VAR IND1,IND2,MIN,PM:INTEGER;
BEGIN
  IF FREQNUL = 0
    THEN
      BEGIN
        FOR IND1:=1 TO DIMPROD DO
          BEGIN
            MIN:=PROD[1];
            PM:=1;
            FOR IND2:=2 TO DIMPROD DO
              IF PROD[IND2] <= MIN THEN
                  BEGIN
                    MIN:=PROD[IND2];
```

```
                    PM:=IND2
                  END;
              PROD[PM]:=MAXINT;
              POSLETCRIT[IND1]:=PM
            END;
          DIMPOSLETCRIT:=DIMPROD
        END
      ELSE
        IF FREQNUL = 1
          THEN
            BEGIN
              POSLETCRIT[1]:=1;
              DIMPOSLETCRIT:=1
            END
          ELSE IF (PROD[FREQNUL-1]=0)AND(PROD[FREQNUL]=0)
              THEN
                BEGIN
                  POSLETCRIT[1]:=FREQNUL-1;
                  POSLETCRIT[2]:=FREQNUL;
                  DIMPOSLETCRIT:=2
                END
              ELSE
                BEGIN
                  POSLETCRIT[1]:=FREQNUL-1;
                  DIMPOSLETCRIT:=1
                END;
END;


PROCEDURE SUPPRESSION(LISTMOT:TABCAR;INDICE:TABIND;DIMMOT,DIMPOSLETCRIT:
                      INTEGER;POSLETCRIT:TAB7;VAR ENTREE:CHAINE;
                      VAR ORTHO:BOOLEAN);
VAR IND,POSCRIT,FUT,BIDON:INTEGER;
    TRAVAIL:CHAINE;
    DISPARU:CHAR;
BEGIN
  IND:=1;
  TRAVAIL:=ENTREE;
  WHILE (IND<=DIMPOSLETCRIT)AND(NOT ORTHO)DO
    BEGIN
      POSCRIT:=POSLETCRIT[IND];
      DELETE(TRAVAIL,POSCRIT,1);
      RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,FUT,BIDON);
      IF NOT ORTHO THEN TRAVAIL:=ENTREE
                   ELSE ENTREE:=TRAVAIL;
      IND:=IND+1
    END
END;

(*$I PROCORMOT *)

   (* PROGRAMME PRINCIPAL *)

BEGIN
 CHARGEMENT(LISTMOT,INDICE,INDICETERM,DIMCAR,DIMMOT,DIMTERM);
 CONSMATDIG(LISTMOT,DIMCAR,DIGRAM);
 CORSTRING;
 IND:=0;
 WRITE ('VOULEZ-VOUS COMMENCER ? (Y/N) : ');
 READLN (REPUTIL);
 WHILE REPUTIL <> 'N' DO
  BEGIN
   IND:=IND+1;
   WRITE ('MOT ',IND,' : ');
   READLN(ENTREE);
   RECHERCHE (LISTMOT,INDICE,DIMMOT,ENTREE,ORTHO,BI,BS);
   IF ORTHO
    THEN WRITELN('ORTHOGRAPHE CORRECTE')
```

```
    ELSE
     BEGIN
      CONTCARCOM(LISTMOT,INDICE,BI,BS,DIMMOT,ENTREE,COMMAX,POSDEPART);
      IF LENGTH(ENTREE)>6
       THEN ESCAPE(LISTMOT,INDICE,INDICETERM,DIMMOT,DIMTERM,POSDEPART,ENTREE,
                   COMMAX,ORTHO)
       ELSE
        BEGIN
         INTERVERTIR(LISTMOT,INDICE,DIMMOT,COMMAX,ENTREE,ORTHO);
         IF NOT ORTHO THEN
          BEGIN
           CALPRODIGMOT(ENTREE,DIGRAM,COMMAX,FREQUENCE,PROD,DIMFREQ,DIMPROD,
               FREQNUL);
           RECHMINPROD(PROD,DIMPROD,FREQNUL,DIMPOSLETCRIT,POSLETCRIT);
           SUPPRESSION(LISTMOT,INDICE,DIMMOT,DIMPOSLETCRIT,POSLETCRIT,ENTREE,
             ORTHO);
           IF NOT ORTHO THEN
            BEGIN
             REMPLACEMENT(LISTMOT,INDICE,DIGRAM,DIMMOT,DIMPOSLETCRIT,
                          POSLETCRIT,ENTREE,ORTHO);
             IF NOT ORTHO THEN INSERTION(LISTMOT,INDICE,DIGRAM,DIMMOT,
                                         DIMFREQ,FREQNUL,FREQUENCE,ENTREE,ORTHO);
            END
          END;
        END;
      IF ORTHO
       THEN WRITELN('PROPOSITION DE CORRECTION:',ENTREE)
       ELSE WRITELN ('JE NE CONNAIS PAS CE MOT')
     END;
    WRITE ('VOULEZ-VOUS CONTINUER ?(O/N)');
    READLN(REPUTIL)
  END
END.
```

```
 (* PROCORMOT *)

PROCEDURE REMPLACLETTRE(POSCRIT:INTEGER;VAR ENTREE:CHAINE;VAR CHOISIE:CHAINE);
BEGIN
  DELETE (ENTREE,POSCRIT,1);
  INSERT (CHOISIE,ENTREE,POSCRIT)
END;

PROCEDURE CLASLETREMP(DIGRAM:MATRICE;LETAV,LETAP:CHAR;VAR LETREMP:TAB27;
                      VAR COMPTLET:INTEGER);
TYPE TABI27 = ARRAY [1..27] OF INTEGER;
VAR VALREMP:TABI27;
    MINCOUR:INTEGER;
    VAL:INTEGER;
    LET:CHAR;
    DERVAL:INTEGER;

PROCEDURE INSERER(VAL:INTEGER;LET:STR1;VAR VALREMP:TABI27;
                  VAR LETREMP:TAB27;VAR MINCOUR:INTEGER);
VAR I:INTEGER;

   PROCEDURE MOVVALLET(DEB,FIN:INTEGER;VAR VALREMP:TABI27;VAR LETREMP:TAB27);
   VAR I:INTEGER;
   BEGIN
     FOR I:= FIN DOWNTO DEB DO
       BEGIN
         LETREMP[I+1]:=LETREMP[I];
         VALREMP[I+1]:=VALREMP[I]
       END
   END;(*MOVVALLET*)

BEGIN
  I:=1;
  WHILE VALREMP[I] > VAL DO I:=I+1;
  MOVVALLET(I,MINCOUR-1,VALREMP,LETREMP);
  LETREMP[I]:=LET;
  VALREMP[I]:=VAL;
  MINCOUR:=MINCOUR+1
END;(*INSERER*)

BEGIN
   DERVAL:=MAXINT;
   MINCOUR:=1;
   FOR LET:='A' TO 'Z' DO
     BEGIN
       VAL:=DIGRAM[LETAV,LET]*DIGRAM[LET,LETAP];
       IF VAL<>0
         THEN
           IF VAL <= DERVAL
           THEN
             BEGIN
               DERVAL:=VAL;
               VALREMP[MINCOUR]:=VAL;
               LETREMP[MINCOUR]:=CORSTR[LET];
               MINCOUR:=MINCOUR+1
             END
           ELSE INSERER(VAL,CORSTR[LET],VALREMP,LETREMP,MINCOUR)
     END;
   COMPTLET:=MINCOUR-1
END;

PROCEDURE REMPLACEMENT(LISTMOT:TABCAR;INDICE:TABIND;DIGRAM:MATRICE;DIMMOT,
                       DIMPOSLETCRIT:INTEGER;POSLETCRIT:TAB7;VAR ENTREE:CHAINE;
                       VAR ORTHO:BOOLEAN);
VAR IND,IND2,BIDON1,BIDON2,COMPTLET,POSCRIT:INTEGER;
    LETREMP:TAB27;
    LETAV,LETAP,LETCA:CHAR;
    TRAVAIL,REMP:CHAINE;
```

```
 PROCEDURE LETAVAP(ENTREE:CHAINE;POSCRIT:INTEGER;
                   VAR LETAV:CHAR;VAR LETAP:CHAR);
 BEGIN
      IF POSCRIT=1 THEN LETAV:='@'
                   ELSE LETAV:=ENTREE[POSCRIT-1];
      IF POSCRIT=LENGTH(ENTREE) THEN LETAP:='@'
                                ELSE LETAP:=ENTREE[POSCRIT+1]
 END;

BEGIN
  IND:=1;
  TRAVAIL:=ENTREE;
  WHILE (IND<=DIMPOSLETCRIT) AND (NOT ORTHO) DO
   BEGIN
    POSCRIT:=POSLETCRIT[IND];
    LETCA:=TRAVAIL[POSCRIT];
    LETAVAP(TRAVAIL,POSCRIT,LETAV,LETAP);
    CLASLETREMP(DIGRAM,LETAV,LETAP,LETREMP,COMPTLET);
    IND2:=1;
    WHILE (IND2<=COMPTLET) AND (NOT ORTHO) DO
     BEGIN
      REMP:=LETREMP[IND2];
      IF (REMP<>CORSTR[LETCA]) THEN
       BEGIN
         REMPLACLETTRE(POSCRIT,TRAVAIL,REMP);
         RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,BIDON1,BIDON2);
         IF (NOT ORTHO) THEN TRAVAIL:=ENTREE
                        ELSE ENTREE:=TRAVAIL
       END;
      IND2:=IND2+1
     END;
    IND:=IND+1;
   END
END;

PROCEDURE INSERTION (LISTMOT:TABCAR;INDICE:TABIND;DIGRAM:MATRICE;DIMMOT,
                     DIMFREQ,FREQNUL:INTEGER;VAR FREQUENCE:TAB7;
                     VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);
VAR IND,IND2,COMPTLET,BIDON1,BIDON2,POSCRIT:INTEGER;
    LETGAU,LETDROI:CHAR;
    INS,TRAVAIL:CHAINE;
    LETINS:TAB27;
    POSDIGCRIT:TAB7;
    DIMPOSDIGCRIT:INTEGER;

 PROCEDURE CLASPOSINS (FREQNUL,DIMFREQ:INTEGER;VAR FREQUENCE,POSDIGCRIT:TAB7;
                       VAR DIMPOSDIGCRIT:INTEGER);
 VAR IND,IND2,MIN,PM:INTEGER;
 BEGIN
  IF FREQNUL <> 0
     THEN
       BEGIN
         IF FREQNUL = 1
            THEN POSDIGCRIT[1]:=1
            ELSE POSDIGCRIT[1]:=FREQNUL;
         DIMPOSDIGCRIT:=1
       END
     ELSE
       BEGIN
         FOR IND:=1 TO DIMFREQ DO
            BEGIN
              MIN:=MAXINT;
              FOR IND2:=1 TO DIMFREQ DO
                BEGIN
                  IF FREQUENCE [IND2] < MIN THEN
                    BEGIN
                      MIN := FREQUENCE[IND2];
```

```
                        PM:=IND2
                      END
                  END;
                FREQUENCE [PM]:=MAXINT;
                POSDIGCRIT[IND]:=PM;
              END;
            DIMPOSDIGCRIT:=DIMFREQ
          END
 END;

 PROCEDURE LETGAUDROI (TRAVAIL:CHAINE;POSCRIT:INTEGER;VAR LETGAU,LETDROI:CHAR);
 BEGIN
   IF POSCRIT=1 THEN LETGAU:='@'
                ELSE LETGAU:=TRAVAIL[POSCRIT-1];
   IF POSCRIT>LENGTH(TRAVAIL) THEN LETDROI:='@'
                              ELSE LETDROI:=TRAVAIL[POSCRIT]
 END;

BEGIN
 CLASPOSINS(FREQNUL,DIMFREQ,FREQUENCE,POSDIGCRIT,DIMPOSDIGCRIT);
 IND:=1;
 TRAVAIL:=ENTREE;
 WHILE(IND<=DIMPOSDIGCRIT)AND(NOT ORTHO) DO
   BEGIN
     POSCRIT:=POSDIGCRIT[IND];
     LETGAUDROI(TRAVAIL,POSCRIT,LETGAU,LETDROI);
     CLASLETREMP(DIGRAM,LETGAU,LETDROI,LETINS,COMPTLET);
     IND2:=1;
     WHILE (IND2<=COMPTLET)AND(NOT ORTHO) DO
       BEGIN
         INS:=LETINS[IND2];
         INSERT(INS,TRAVAIL,POSCRIT);
         RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,BIDON1,BIDON2);
         IF NOT ORTHO THEN TRAVAIL:=ENTREE
                      ELSE ENTREE:=TRAVAIL;
         IND2:=IND2 + 1
       END;
     IND:=IND + 1
   END
END;

PROCEDURE INTERVERTIR(LISTMOT:TABCAR;INDICE:TABIND;DIMMOT,COMMAX:INTEGER;
                      VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);
VAR INTER:CHAR;
    IND,BIDON1,BIDON2:INTEGER;
    STRLET,TRAVAIL:CHAINE;
BEGIN
 IF LENGTH(ENTREE)>=2 THEN
  BEGIN
   IND:=1;
   TRAVAIL:=ENTREE;
   WHILE (NOT ORTHO) AND (IND<=COMMAX+1) AND (IND < LENGTH(TRAVAIL)) DO
     BEGIN
       INTER:=TRAVAIL[IND+1];
       DELETE(TRAVAIL,IND+1,1);
       INSERT(CORSTR[INTER],TRAVAIL,IND);
       RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,BIDON1,BIDON2);
       IF NOT ORTHO THEN TRAVAIL:=ENTREE
                    ELSE ENTREE:=TRAVAIL;
       IND:=IND+1
     END
  END
END;

PROCEDURE ESCAPE(LISTMOT:TABCAR;INDICE:TABIND;INDICETERM:TABTERM;DIMMOT,
                 DIMTERM,POSDEPART:INTEGER;VAR ENTREE:CHAINE;
                 VAR COMMAX:INTEGER;VAR ORTHO:BOOLEAN);
```

```
VAR BI,BS,LONGTERM,DEBTERMENT,DEMI,INDCAR,DISTMIN,DISTCAND:INTEGER;
    TROUVE,CONTINUE:BOOLEAN;
    TERMENT,SUFFIXE,INTER:CHAINE;

PROCEDURE DISTANCE(ENTREE,STR:CHAINE;DISTMIN:INTEGER;VAR DIST:INTEGER);
VAR LONG,IND,IND2:INTEGER;
    STRBIS:CHAINE;
    RESULTAT:BOOLEAN;

 PROCEDURE INTERCHANGER(VAR RESULTAT:BOOLEAN);
 BEGIN
  RESULTAT:=FALSE;
  IF ((IND2+1) <= LENGTH(STRBIS)) AND ((IND+1) <= LENGTH(ENTREE))
    THEN IF (ENTREE[IND]=STRBIS[IND2+1]) AND (ENTREE[IND+1]=STRBIS[IND2])
           THEN
            BEGIN
             DIST:=DIST+1;
             IND:=IND+2;
             IND2:=IND2+2;
             RESULTAT:=TRUE
            END
 END;

 PROCEDURE SUBSTITUER(VAR RESULTAT:BOOLEAN);
 BEGIN
  RESULTAT:=FALSE;
  IF((IND+1) <= LENGTH(ENTREE)) AND ((IND2+1) <= LENGTH(STRBIS))
    THEN IF ENTREE[(IND+1)]=STRBIS[(IND2+1)]
      THEN
        BEGIN
          DIST:=DIST+1;
          IND:=IND+2;
          IND2:=IND2+2;
          RESULTAT:=TRUE
        END
 END;

 PROCEDURE AJOUTER(VAR RESULTAT:BOOLEAN);
 BEGIN
  RESULTAT:=FALSE;
  IF (IND+1) <= LENGTH(ENTREE)
    THEN IF ENTREE[IND+1]=STRBIS[IND2]
      THEN
        BEGIN
          DIST:=DIST+1;
          LONG:=LONG+1;
          IND:=IND+2;
          IND2:=IND2+1;
          RESULTAT:=TRUE
        END
 END;

 PROCEDURE OUBLIER(VAR RESULTAT:BOOLEAN);
 BEGIN
   RESULTAT:=FALSE;
   IF (IND2+1) <= LENGTH(STRBIS)
     THEN IF ENTREE[IND]=STRBIS[IND2+1]
       THEN
         BEGIN
           DIST:=DIST+1;
           LONG:=LONG-1;
           IND:=IND+1;
           IND2:=IND2+2;
           RESULTAT:=TRUE
         END
 END;

 (* DEBUT PROCEDURE DISTANCE *)
```

```
BEGIN
 STRBIS:=STR;
 LONG:=LENGTH(STRBIS);
 DIST:=0;
 IND:=1;
 IND2:=1;
 WHILE (IND<=LENGTH(ENTREE)) AND (IND2<=LENGTH(STRBIS)) AND (DIST<DISTMIN) DO
  BEGIN
   IF ENTREE[IND]=STRBIS[IND2]
    THEN
     BEGIN
      IND:=IND+1;
      IND2:=IND2+1
     END
    ELSE
     BEGIN
      INTERCHANGER(RESULTAT);
      IF NOT RESULTAT
       THEN
        BEGIN
         SUBSTITUER(RESULTAT);
         IF NOT RESULTAT
          THEN
           BEGIN
            AJOUTER(RESULTAT);
            IF NOT RESULTAT
             THEN
              BEGIN
               OUBLIER(RESULTAT);
               IF NOT RESULTAT
                THEN
                 BEGIN
                  IF(IND=LENGTH(ENTREE))OR(IND2=LENGTH(STRBIS))
                      THEN DIST:=DIST+1
                      ELSE DIST:=DIST+2;
                  IND:=IND+1;
                  IND2:=IND2+1
                 END
              END
        END
     END
    END
  END;
 DIST:=DIST+ABS(LENGTH(ENTREE)-LONG)
END;

PROCEDURE LISTCANDIDAT(ENTREE:CHAINE;COMMAX,POSDEPART:INTEGER;
                       VAR DISTMIN:INTEGER;VAR INTER:CHAINE);
VAR IND2,MAXCOM,IND3,LONGUEUR:INTEGER;
    MEILLEUR,SINTER:CHAINE;
BEGIN
 MAXCOM:=COMMAX;
 IND2:=POSDEPART;
 WHILE(MAXCOM=COMMAX)AND(IND2>0)AND(DISTMIN>1) DO
   BEGIN
     IND3:=INDICE[IND2];
     LONGUEUR:=INDICE[IND2+1]-IND3-1;
     CHARBYTE(LISTMOT,IND3,LONGUEUR,SINTER);
     COMPARE(ENTREE,SINTER,MAXCOM);
     IF COMMAX=MAXCOM THEN
       BEGIN
         IND2:=IND2-1;
         DISTANCE(ENTREE,SINTER,DISTMIN,DISTCAND);
         IF DISTCAND<DISTMIN THEN
                              BEGIN
                                MEILLEUR:=SINTER;
                                DISTMIN:=DISTCAND
```

```
                                    END
        END
    END;
  IND2:=POSDEPART+1;
  MAXCOM:=COMMAX;
  WHILE(MAXCOM=COMMAX)AND(IND2<316)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICE[IND2];
      LONGUEUR:=INDICE[IND2+1]-IND3-1;
      CHARBYTE(LISTMOT,IND3,LONGUEUR,SINTER);
      COMPARE(ENTREE,SINTER,MAXCOM);
      IF COMMAX=MAXCOM THEN
        BEGIN
          IND2:=IND2+1;
          DISTANCE(ENTREE,SINTER,DISTMIN,DISTCAND);
          IF DISTCAND<DISTMIN THEN
                                    BEGIN
                                      MEILLEUR:=SINTER;
                                      DISTMIN:=DISTCAND
                                    END
        END
    END;
    INTER:=MEILLEUR
  END;

PROCEDURE LISTSUFFCANDI(SUFFIXE:CHAINE;POSDEPART:INTEGER;VAR DISTMIN:INTEGER;
                        VAR INTER:CHAINE);
VAR IND2,IND3,LONGMOT,DEBTERM:INTEGER;
    MOT,SUFFCANDI:CHAINE;
    CH:CHAR;
BEGIN
  IND2:=POSDEPART;
  SUFFCANDI:=SUFFIXE;
  WHILE(SUFFCANDI=SUFFIXE)AND(IND2>0)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICETERM[IND2];
      CHARBYTE(LISTMOT,IND3,LONGTERM,SUFFCANDI);
      IF SUFFCANDI=SUFFIXE THEN
        BEGIN
          IND2:=IND2-1;
          CH:=LISTMOT[IND3];
          LONGMOT:=2;
          WHILE (CH<>'@')AND(IND3>0)DO
            BEGIN
              IND3:=IND3-1;
              LONGMOT:=LONGMOT+1;
              CH:=LISTMOT[IND3]
            END;
          IND3:=IND3+1;
          CHARBYTE(LISTMOT,IND3,LONGMOT,MOT);
          DISTANCE(ENTREE,MOT,DISTMIN,DISTCAND);
          IF DISTCAND<DISTMIN THEN
            BEGIN
              INTER:=MOT;
              DISTMIN:=DISTCAND
            END
        END
    END;
  IND2:=POSDEPART+1;
  SUFFCANDI:=SUFFIXE;
  WHILE(SUFFCANDI=SUFFIXE)AND(IND2<99)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICETERM[IND2];
      CHARBYTE(LISTMOT,IND3,LONGTERM,SUFFCANDI);
      IF SUFFCANDI=SUFFIXE THEN
        BEGIN
          IND2:=IND2+1;
          CH:=LISTMOT[IND3];
```

```
              LONGMOT:=2;
              WHILE (CH<>'@')AND(IND3>0)DO
                BEGIN
                  IND3:=IND3-1;
                  LONGMOT:=LONGMOT+1;
                  CH:=LISTMOT[IND3];
                END;
                 IND3:=IND3+1;
                 CHARBYTE(LISTMOT,IND3,LONGMOT,MOT);
                 DISTANCE(ENTREE,MOT,DISTMIN,DISTCAND);
                 IF DISTCAND<DISTMIN THEN
                   BEGIN
                     INTER:=MOT;
                     DISTMIN:=DISTCAND
                   END
              END
          END
    END;

(* PROCEDURE ESCAPE  *)

BEGIN
  DISTMIN:=MAXINT;
  LONGTERM:=3;
  IF COMMAX>4 THEN
                BEGIN
                   LISTCANDIDAT(ENTREE,COMMAX,POSDEPART,DISTMIN,INTER);
                   ORTHO:=TRUE;
                   ENTREE:=INTER
                END;
  IF DISTMIN=1
     THEN CONTINUE:=FALSE
     ELSE CONTINUE:=TRUE;
  BI:=1;
  BS:=DIMTERM;
  DEBTERMENT:=LENGTH(ENTREE)-2;
  TERMENT:=COPY(ENTREE,DEBTERMENT,3);
  WHILE CONTINUE AND(BI<=BS)DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICETERMI[DEMI];
      CHARBYTE(LISTMOT,INDCAR,LONGTERM,SUFFIXE);
      IF TERMENT = SUFFIXE
        THEN
          BEGIN
            CONTINUE:=FALSE;
            LISTSUFFCANDI(TERMENT,DEMI,DISTMIN,INTER);
            ORTHO:=TRUE;
            ENTREE:=INTER;
          END
        ELSE IF TERMENT < SUFFIXE
          THEN BS:=DEMI-1
          ELSE BI:=DEMI+1
    END
END;
```

A N N E X E 6,

MATRICE DES FREQUENCES DES DIGRAMMES DU DICTIONNAIRE DE LA CORRECTION D'ORTHOGRAPHE.

-	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Á	É	Í	Ó	Ú	Ü	Ñ	
	17	8	43	14	17	15	7	8	3	6	2	9	27	9	5	50		15	23	24		11				4								-
108		3	10	16	2	2	3		2	5		17	8	20		6	1	31	10	12	2	3			1	6							6	A
	4				1				4			2			5			14			1									1	1			B
	28		1		5			5	25						20					3	7						1	1	1	4				C
8	13				6				13						12			2			2							1	1					D
32	2	2	7	8		3	3		1	5		10	5	20	3			31	28	10		2				4							3	E
	5				5				3			1			2			4			1						1				1			F
	5				2				3			1			3			2			2											1		G
	3				2				1						4						3													H
	12	2	11	7	13	1	2					11	3	22	11	1		3	12	10	1					1				12			1	I
1	8				5										5						1							1						J
									2																									K
8	18		1	2	10				5			12	1		14				1		5						1		1					L
1	19	5			13				7						7	5					2						1	1		1	1			M
24	17		5		7	1	1		4	1					16				3	19	1	1				2					1			N
105		6	9	1		1				2		9	10	6		3		22	7	3		2			1									O
	23				16				7			3			8			7			6						2							P
																					5													Q
16	26	3	3	4	21				13			1	3	2	27		2	8	1	13	5	1				1	1	1	2	1				R
9	16		7		12				13						7	7	1			11	4	1								3				S
	37	1			18				14						26			7			3								1	3				T
	4	2	1	4	16		1		4	1		3	2					7	2	3		1				2		1						U
	2				9				9						1												1							V
																																		W
									1																									X
	2																																	Y
																																		Z
4	8						1								3						3									1				

A6.2

-	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Á	É	Í	Ó	Ú	Ü	Ñ
		1	1				1			1				1			1	1	1														
			1	1										1	1			1	1					1								
	3		1												1																	
				1									3	20						1												
													1						2	1												
					1																											
	3								1						5																	

Á É Í Ó Ú Ü Ñ

A6.3

A N N E X E 7.

SPECIFICATIONS DE CORESPLUR.

(Correction d'orthographe de mots singuliers et pluriels).

Voir ANNEXE 4 sauf les spécifications de CORMOT.

Programme CORMOTPLUR.

Argument : - ENTREE : mot espagnol entré au terminal sur proposition du programme singulier ou pluriel.

Résultat : dit si "ENTREE", au singulier ou au pluriel, est correctement orthographié, sinon propose une correction. Si seule la formation du pluriel est incorrecte, la correction proposée est la forme correcte du mot au pluriel, sinon propose une correction qui est un mot appartenant au dictionnaire de référence du programme et qui est

- si "ENTREE" compte plus de 6 lettres, le mot qui en est le plus proche selon la procédure "ESCAPE".
- si "ENTREE" ne compte pas plus de 6 lettres, un mot obtenu après une modification de "ENTREE" telle que interversion de lettres, suppression de lettre, remplacement de lettre ou insertion de lettre. Cette correction sera mise correctement au pluriel si "ENTREE" l'était.
  Si le programme ne parvient pas à proposer une correction, il répond qu'il ne connaît pas "ENTREE".

Fonction : le programme reçoit des mots un par un au terminal. Pour chaque mot, il vérifie s'il se trouve dans son dictionnaire de référence. Si oui, il répond que l'orthographe est correcte. Si non, il essaie de le corriger de la manière suivante.

Tout d'abord, il essaie de mettre le mot au singulier.
Si la forme singulier est dans le dictionnaire et que la formation du pluriel est correcte, le module répond que l'orthographe est exacte.
Si la forme singulier est dans le dictionnaire mais la forme du pluriel n'est pas exacte, le module propose comme correction la forme pluriel correcte.
Si la forme singulier n'est pas dans le dictionnaire, le module essaie de la corriger de la manière suivante.
Si me mot au singulier compte plus de 6 lettres, il fait appel à la procédure "ESCAPE" et proposera le mot donné par celle-ci comme solution.
Sinon, il modifie le mot jusqu'à ce que le mot ainsi construit soit dans le dictionnaire ou jusqu'à ce qu'il arrive au bout des opérations suivantes en travaillant sur le plus grand nombre de lettres initiales que le mot a en commun avec le dictionnaire : interversion de lettres, suppression de lettre, remplacement de lettre et insertion de lettre.

Procédure MISESING.

Arguments : - ENTREE : string, mot que l'on traite.

Résultats : - ORTHO : vrai si "ENTREE" est dans le dictionnaire, faux sinon.
- PLURIEL : vrai si "ENTREE" est au pluriel à son entrée dans la procédure, faux sinon.
- FORMPLUR : vrai si "ENTREE" est au pluriel à son entrée dans la procédure et que le pluriel est correctement formé, faux si le pluriel n'est pas correctement formé.

Fonction : - si "ENTREE" est au pluriel, met"PLURIEL"à vrai et met "ENTREE" au singulier.
- si le pluriel est correctement formé, met "FORMPLUR" à vrai, sinon "FORMPLUR" est mis à faux.
- si la forme singulier est dans le dictionnaire, met "ORTHO" à vrai, sinon "ORTHO" est faux.

Procédure MISEPLUR.

Arguments : - PLURIEL : vrai s'il faut mettre "ENTREE" au pluriel, faux sinon.
- ENTREE : string, mot espagnol à mettre au pluriel.

Résultat : - ENTREE : string, mot espagnol mis au pluriel.

Fonction : met "ENTREE", mot espagnol, au pluriel si "PLURIEL" est vrai selon les règles de grammaire espagnole : ajoute "s" si la dernière lettre de "ENTREE" au singulier est une voyelle, ajoute "es" si la dernière lettre de "ENTREE" est une consonne.
Si "ENTREE" est terminé par "z" au singulier, remplace ce "z" par "ces" et si "ENTREE" compte plus de 4 lettres et se termine par "s", ne le modifie pas.

```
(* ANNEXE 8:MODULE DE CORRECTION AUTOMATIQUE D'ORTHOGRAPHE DE MOTS
                ESPAGNOLS SINGULIERS ET PLURIELS.
  ================================================================*)

(*$S+*)
PROGRAM CORESPLUR;
(*================*)
CONST NBREMOT = 320;
      NBRECAR = 2200;
      NBRETERM =100;
TYPE MATRICE=ARRAY ['@'..'Z','@'..'Z'] OF INTEGER;
     FICHIER = FILE OF STRING;
     CHAINE=STRING[20];
     TABCAR= PACKED ARRAY[1..NBRECAR] OF CHAR;
     TAB7= ARRAY [1..7] OF INTEGER;
     TAB27= ARRAY [1..27] OF CHAINE;
     TABIND= ARRAY[1..NBREMOT] OF INTEGER;
     TABTERM= ARRAY[1..NBRETERM]OF INTEGER;
     STR1=STRING[1];
VAR REPUTIL:CHAR;
    BI,BS,DIMMOT,DIMCAR,DIMTERM,IND,COMMAX,POSDEPART,FREQNUL,DIMPROD,DIMFREQ,
      DIMPOSLETCRIT:INTEGER;
    ENTREE:CHAINE;
    FORMPLUR,PLURIEL,ORTHO:BOOLEAN;
    POSLETCRIT,PROD,FREQUENCE:TAB7;
    DIGRAM:MATRICE;
    INDICE:TABIND;
    INDICETERM:TABTERM;
    LISTMOT:TABCAR;
    DICOSUBST:FICHIER;
    CORSTR:PACKED ARRAY['@'..'Z']OF STR1;

PROCEDURE CHARGEMENT(VAR LISTMOT:TABCAR;VAR INDICE:TABIND;VAR INDICETERM:
                     TABTERM;VAR DIMCAR,DIMMOT,DIMTERM:INTEGER);
VAR IND,IND3,IND4,IND5,DEBTERMENT,INDTERMENT:INTEGER;
    TERMENT:STRING[3];
    TABTERM:ARRAY[1..NBREMOT]OF STRING[3];
BEGIN
  WRITELN('DEBUT DE CHARGEMENT,DUREE:20 SECONDES!');
  IND3:=0;
  IND:=1;
  TABTERM[1]:='ZZZ';
  LISTMOT[IND]:='@';
  DIMTERM:=0;
  RESET(DICOSUBST,'#5:DICTION.DATA');
  WHILE NOT EOF(DICOSUBST) DO
     BEGIN
       IND3:=IND3+1;
       IND:=IND+1;
       INDICE[IND3]:=IND;
       ENTREE:=DICOSUBST^;
       MOVELEFT(ENTREE[1],LISTMOT[IND],LENGTH(ENTREE));
       IND:=IND+LENGTH(ENTREE);
       IF LENGTH(ENTREE) >6 THEN
         BEGIN
           DEBTERMENT:=LENGTH(ENTREE)-2;
           TERMENT:=COPY(ENTREE,DEBTERMENT,3);
           INDTERMENT:=IND-3;
           IND4:=1;
           WHILE ((IND4<=DIMTERM)AND(TERMENT>=TABTERM[IND4])) DO IND4:=IND4+1;
           DIMTERM:=DIMTERM+1;
           IF DIMTERM>1 THEN FOR IND5:=DIMTERM DOWNTO IND4+1 DO
             BEGIN
               TABTERM[IND5]:=TABTERM[IND5-1];
               INDICETERM[IND5]:=INDICETERM[IND5-1]
             END;
           TABTERM[IND4]:=TERMENT;
           INDICETERM[IND4]:=INDTERMENT
```

```
          END;
        LISTMOT[IND]:='@';
        GET(DICOSUBST)
      END;
  INDICE[IND3+1]:=IND+1;
  LISTMOT[IND+1]:='Z';
  LISTMOT[IND+2]:='Z';
  LISTMOT[IND+3]:='@';
  DIMCAR:=IND+3;
  DIMMOT:=IND3+1;
  WRITELN('DIMENSION DE LISTMOT:',DIMCAR);
  WRITELN('NOMBRE DE MOTS:',DIMMOT);
  CLOSE(DICOSUBST,LOCK)
END;

PROCEDURE CORSTRING;
VAR I:CHAR;
BEGIN
    CORSTR['@']:='@';
    CORSTR['A']:='A';
    CORSTR['B']:='B';
    CORSTR['C']:='C';
    CORSTR['D']:='D';
    CORSTR['E']:='E';
    CORSTR['F']:='F';
    CORSTR['G']:='G';
    CORSTR['H']:='H';
    CORSTR['I']:='I';
    CORSTR['J']:='J';
    CORSTR['K']:='K';
    CORSTR['L']:='L';
    CORSTR['M']:='M';
    CORSTR['N']:='N';
    CORSTR['O']:='O';
    CORSTR['P']:='P';
    CORSTR['Q']:='Q';
    CORSTR['R']:='R';
    CORSTR['S']:='S';
    CORSTR['T']:='T';
    CORSTR['U']:='U';
    CORSTR['V']:='V';
    CORSTR['W']:='W';
    CORSTR['X']:='X';
    CORSTR['Y']:='Y';
    CORSTR['Z']:='Z';
END;

PROCEDURE CONSMATDIG(LISTMOT:TABCAR;DIMCAR:INTEGER;VAR DIGRAM:MATRICE);
VAR LIGNE,COLONNE:CHAR;
    I:INTEGER;
BEGIN
  WRITELN('CHARGEMENT DE LA MATRICE.');
  FOR LIGNE:= '@' TO 'Z' DO
    FOR COLONNE:= '@' TO 'Z' DO
      DIGRAM[LIGNE,COLONNE]:=0;
  FOR I:=1 TO DIMCAR-4 DO
    DIGRAM[LISTMOT[I],LISTMOT[I+1]]:=DIGRAM[LISTMOT[I],LISTMOT[I+1]] + 1
END;

PROCEDURE CHARBYTE(LISTMOT:TABCAR;SOURCE:INTEGER;VAR LONGUEUR:INTEGER;
                   VAR DESTINATION:CHAINE);
BEGIN
  LONGUEUR:=LONGUEUR+1;
  MOVELEFT(LISTMOT[SOURCE-1],DESTINATION,LONGUEUR);
  LONGUEUR:=LONGUEUR-1;
  MOVELEFT(LONGUEUR,DESTINATION,1)
END;
```

```
PROCEDURE COMPARE(ENTREE,FICHMOT:CHAINE;VAR COMCOUR :INTEGER);
VAR TROUVE:BOOLEAN;
    IND:INTEGER;
BEGIN
  TROUVE:=TRUE;IND:=1;
  WHILE((IND<=LENGTH(ENTREE))AND(IND<=(LENGTH(FICHMOT)))AND TROUVE)
    DO IF ENTREE[IND] = FICHMOT[IND]
      THEN IND:=IND+1
      ELSE TROUVE:=FALSE;
  COMCOUR:=IND-1
END;

PROCEDURE RECHERCHE (LISTMOT:TABCAR;INDICE:TABIND;DIMMOT:INTEGER;ENTREE:CHAINE;
                     VAR ORTHO:BOOLEAN;VAR BI,BS:INTEGER);
VAR DEMI,INDCAR,LONGUEUR:INTEGER;
    INTER:CHAINE;
    CONTINUE:BOOLEAN;
BEGIN
  ORTHO:=FALSE;
  CONTINUE:=TRUE;
  BI:=1;
  BS:=DIMMOT-1;
  WHILE (CONTINUE)AND(BI<=BS) DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICE[DEMI];
      LONGUEUR:=INDICE[DEMI+1]-INDCAR-1;
      CHARBYTE(LISTMOT,INDCAR,LONGUEUR,INTER);
      IF ENTREE=INTER
            THEN
             BEGIN
              CONTINUE:=FALSE;
              ORTHO:=TRUE
             END
            ELSE IF ENTREE<INTER
                 THEN BS:=DEMI-1
                 ELSE BI:=DEMI+1
    END
 END;(*RECHERCHE*)

 PROCEDURE CONTCARCOM(LISTMOT:TABCAR;INDICE:TABIND;BI,BS,DIMMOT:INTEGER;
                      ENTREE:CHAINE;VAR COMMAX,POSDEPART:INTEGER);
 VAR COMCOUR:INTEGER;

   PROCEDURE COMPMOT(NO:INTEGER;ENTREE:CHAINE;VAR COMMUNE:INTEGER);
   VAR MOT:CHAINE;
       LONGUEUR:INTEGER;
   BEGIN
      LONGUEUR:=INDICE[NO+1]-INDICE[NO]-1;
      CHARBYTE(LISTMOT,INDICE[NO],LONGUEUR,MOT);
      COMPARE(ENTREE,MOT,COMMUNE)
   END;

 BEGIN
  COMCOUR:=0;
  COMMAX:=0;
  IF BS>0 THEN COMPMOT(BS,ENTREE,COMCOUR);
  IF BI<(DIMMOT-1)THEN
                    BEGIN
                      COMPMOT(BI,ENTREE,COMMAX);
                      POSDEPART:=BI
                    END;
  IF COMCOUR>COMMAX
     THEN
        BEGIN
           COMMAX:=COMCOUR;
           POSDEPART:=BS
        END
     ELSE POSDEPART:=BI
END;(*CONTCARCOM*)
```

```
PROCEDURE MISESING(VAR ENTREE:CHAINE;VAR ORTHO,PLURIEL,FORMPLUR:BOOLEAN);
VAR VOYELLES:SET OF 'A'..'U';
    FUT,BIDON:INTEGER;
BEGIN
 VOYELLES:=['A','E','I','O','U'];
 IF ENTREE[LENGTH(ENTREE)]='S'
  THEN
   BEGIN
    PLURIEL:=TRUE;
    DELETE(ENTREE,LENGTH(ENTREE),1);
    RECHERCHE(LISTMOT,INDICE,DIMMOT,ENTREE,ORTHO,FUT,BIDON);
    IF NOT ORTHO
     THEN
      BEGIN
       IF ENTREE[LENGTH(ENTREE)]='E'
        THEN
         BEGIN
          DELETE(ENTREE,LENGTH(ENTREE),1);
          IF ENTREE[LENGTH(ENTREE)]='C'
           THEN
            BEGIN
             DELETE(ENTREE,LENGTH(ENTREE),1);
             ENTREE:=CONCAT(ENTREE,'Z');
             FORMPLUR:=TRUE
            END
           ELSE IF ENTREE[LENGTH(ENTREE)]='Z'
                   THEN FORMPLUR:=FALSE
                   ELSE IF(ENTREE[LENGTH(ENTREE)]='S')AND(LENGTH(ENTREE)>4)
                           THEN FORMPLUR:=FALSE
                           ELSE  FORMPLUR:=TRUE;
          RECHERCHE(LISTMOT,INDICE,DIMMOT,ENTREE,ORTHO,FUT,BIDON);
          IF ORTHO
           THEN IF ENTREE[LENGTH(ENTREE)] IN VOYELLES
                   THEN FORMPLUR:=FALSE
         END
      END
       ELSE IF NOT(ENTREE[LENGTH(ENTREE)] IN VOYELLES)
              THEN FORMPLUR:=FALSE
              ELSE FORMPLUR:=TRUE
   END
  ELSE
   BEGIN
    PLURIEL:=FALSE;
    FORMPLUR:=TRUE
   END
END;(*MISESING*)

PROCEDURE CALPRODIGMOT(ENTREE:CHAINE;DIGRAM:MATRICE;COMMAX:INTEGER;
                       VAR FREQUENCE,PROD:TAB7;VAR DIMFREQ,DIMPROD,FREQNUL:INTEGER);
VAR PREMIND,DEUXIND:CHAR;
    TRAVAIL:CHAINE;
    CONTINUE:BOOLEAN;
    LGTRAV,IND:INTEGER;
BEGIN
  IF(LENGTH(ENTREE)=COMMAX) OR (LENGTH(ENTREE)=COMMAX+1)
    THEN
      BEGIN
        TRAVAIL:=CONCAT('@',ENTREE,'@');
        LGTRAV:=LENGTH(TRAVAIL)
      END
    ELSE
      BEGIN
        TRAVAIL:=CONCAT('@',ENTREE);
        LGTRAV:=COMMAX+3
      END;
  CONTINUE:=TRUE;
  FREQNUL:=0;
  IND:=1;
```

```
  DIMPROD:=0;
  DIMFREQ:=0;
  WHILE (IND<LGTRAV) AND CONTINUE DO
    BEGIN
      PREMIND:=TRAVAIL[IND];
      DEUXIND:=TRAVAIL[IND+1];
      DIMFREQ:=DIMFREQ+1;
      FREQUENCE[DIMFREQ]:=DIGRAM[PREMIND,DEUXIND];
      IF DIMFREQ>1
        THEN
          BEGIN
            DIMPROD:=DIMPROD+1;
            PROD[DIMPROD]:=FREQUENCE[DIMFREQ-1]*FREQUENCE[DIMFREQ]
          END;
      IF FREQUENCE[DIMFREQ]=0
        THEN
          BEGIN
            IF IND <> LGTRAV-1
              THEN
                BEGIN
                  DIMPROD:=DIMPROD+1;
                  PROD[DIMPROD]:=0
                END;
            FREQNUL:=IND;
            CONTINUE:=FALSE
          END;
      IND:=IND + 1;
    END
END;

PROCEDURE RECHMINPROD(PROD:TAB7;DIMPROD,FREQNUL:INTEGER;
                      VAR DIMPOSLETCRIT:INTEGER;VAR POSLETCRIT:TAB7);

VAR IND1,IND2,MIN,PM:INTEGER;
BEGIN
  IF FREQNUL = 0
    THEN
      BEGIN
        FOR IND1:=1 TO DIMPROD DO
          BEGIN
            MIN:=PROD[1];
            PM:=1;
            FOR IND2:=2 TO DIMPROD DO
              IF PROD[IND2] <= MIN THEN
                  BEGIN
                    MIN:=PROD[IND2];
                    PM:=IND2
                  END;
            PROD[PM]:=MAXINT;
            POSLETCRIT[IND1]:=PM
          END;
        DIMPOSLETCRIT:=DIMPROD
      END
    ELSE
      IF FREQNUL = 1
        THEN
          BEGIN
            POSLETCRIT[1]:=1;
            DIMPOSLETCRIT:=1
          END
        ELSE IF (PROD[FREQNUL-1]=0)AND(PROD[FREQNUL]=0)
            THEN
              BEGIN
                POSLETCRIT[1]:=FREQNUL-1;
                POSLETCRIT[2]:=FREQNUL;
                DIMPOSLETCRIT:=2
              END
            ELSE
```

```
                BEGIN
                  POSLETCRIT[1]:=FREQNUL-1;
                  DIMPOSLETCRIT:=1
                END;
END;


PROCEDURE SUPPRESSION(LISTMOT:TABCAR;INDICE:TABIND;DIMMOT,DIMPOSLETCRIT:
                      INTEGER;POSLETCRIT:TAB7;VAR ENTREE:CHAINE;
                      VAR ORTHO:BOOLEAN);
VAR IND,POSCRIT,FUT,BIDON:INTEGER;
    TRAVAIL:CHAINE;
    DISPARU:CHAR;
BEGIN
  IND:=1;
  TRAVAIL:=ENTREE;
  WHILE (IND<=DIMPOSLETCRIT)AND(NOT ORTHO)DO
    BEGIN
      POSCRIT:=POSLETCRIT[IND];
      DELETE(TRAVAIL,POSCRIT,1);
      RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,FUT,BIDON);
      IF NOT ORTHO THEN TRAVAIL:=ENTREE
                   ELSE ENTREE:=TRAVAIL;
      IND:=IND+1
    END
END;


(*$I CORESPLUR2 *)

   (* PROGRAMME PRINCIPAL *)

BEGIN
 CHARGEMENT(LISTMOT,INDICE,INDICETERM,DIMCAR,DIMMOT,DIMTERM);
 CONSMATDIG(LISTMOT,DIMCAR,DIGRAM);
 CORSTRING;
 IND:=0;
 WRITE ('VOULEZ-VOUS COMMENCER ? (Y/N) : ');
 READLN (REPUTIL);
 WHILE REPUTIL <> 'N' DO
  BEGIN
   IND:=IND+1;
   WRITE ('MOT ',IND,' : ');
   READLN(ENTREE);
   RECHERCHE (LISTMOT,INDICE,DIMMOT,ENTREE,ORTHO,BI,BS);
   IF ORTHO
    THEN WRITELN('ORTHOGRAPHE CORRECTE')
    ELSE
     BEGIN
      MISESING(ENTREE,ORTHO,PLURIEL,FORMPLUR);
      IF NOT ORTHO
       THEN
        BEGIN
          CONTCARCOM(LISTMOT,INDICE,BI,BS,DIMMOT,ENTREE,COMMAX,POSDEPART);
          IF LENGTH(ENTREE)>6
           THEN ESCAPE(LISTMOT,INDICE,INDICETERM,DIMMOT,DIMTERM,POSDEPART,
                       ENTREE,COMMAX,ORTHO)
           ELSE
            BEGIN
              INTERVERTIR(LISTMOT,INDICE,DIMMOT,COMMAX,ENTREE,ORTHO);
              IF NOT ORTHO THEN
```

```
                BEGIN
                 CALPRODIGMOT(ENTREE,DIGRAM,COMMAX,FREQUENCE,PROD,DIMFREQ,
                     DIMPROD,FREQNUL);
                 RECHMINPROD(PROD,DIMPROD,FREQNUL,DIMPOSLETCRIT,POSLETCRIT);
                 SUPPRESSION(LISTMOT,INDICE,DIMMOT,DIMPOSLETCRIT,POSLETCRIT,
                             ENTREE,ORTHO);
                 IF NOT ORTHO THEN
                  BEGIN
                   REMPLACEMENT(LISTMOT,INDICE,DIGRAM,DIMMOT,DIMPOSLETCRIT,
                                POSLETCRIT,ENTREE,ORTHO);
                    IF NOT ORTHO THEN INSERTION(LISTMOT,INDICE,DIGRAM,DIMMOT,
                                                DIMFREQ,FREQNUL,FREQUENCE,ENTREE,
                                                ORTHO);
                  END
                END
              END;
            IF ORTHO
             THEN
              BEGIN
               MISEPLUR(PLURIEL,ENTREE);
               WRITELN('PROPOSITION DE CORRECTION:',ENTREE)
              END
             ELSE WRITELN ('JE NE CONNAIS PAS CE MOT')
           END
          ELSE IF FORMPLUR THEN WRITELN('ORTHOGRAPHE CORRECTE')
                           ELSE
                            BEGIN
                             MISEPLUR(PLURIEL,ENTREE);
                             WRITELN('MAUVAISE FORMATION DU PLURIEL!');
                             WRITELN('       PLURIEL CORRECT:',ENTREE)
                            END
      END;
    WRITE ('VOULEZ-VOUS CONTINUER ?(O/N)');
    READLN(REPUTIL)
  END
END.
```

```
 (* CORESPLUR2 *)

PROCEDURE REMPLACLETTRE(POSCRIT:INTEGER;VAR ENTREE:CHAINE;VAR CHOISIE:CHAINE);
BEGIN
  DELETE (ENTREE,POSCRIT,1);
  INSERT (CHOISIE,ENTREE,POSCRIT)
END;

PROCEDURE CLASLETREMP(DIGRAM:MATRICE;LETAV,LETAP:CHAR;VAR LETREMP:TAB27;
                      VAR COMPTLET:INTEGER);
TYPE TABI27 = ARRAY [1..27] OF INTEGER;
VAR VALREMP:TABI27;
    MINCOUR:INTEGER;
    VAL:INTEGER;
    LET:CHAR;
    DERVAL:INTEGER;

PROCEDURE INSERER(VAL:INTEGER;LET:STR1;VAR VALREMP:TABI27;
                  VAR LETREMP:TAB27;VAR MINCOUR:INTEGER);
VAR I:INTEGER;

   PROCEDURE MOVVALLET(DEB,FIN:INTEGER;VAR VALREMP:TABI27;VAR LETREMP:TAB27);
   VAR I:INTEGER;
   BEGIN
     FOR I:= FIN DOWNTO DEB DO
       BEGIN
         LETREMP[I+1]:=LETREMP[I];
         VALREMP[I+1]:=VALREMP[I]
       END
   END;(*MOVVALLET*)

BEGIN
  I:=1;
  WHILE VALREMP[I] > VAL DO I:=I+1;
  MOVVALLET(I,MINCOUR-1,VALREMP,LETREMP);
  LETREMP[I]:=LET;
  VALREMP[I]:=VAL;
  MINCOUR:=MINCOUR+1
END;(*INSERER*)

BEGIN
   DERVAL:=MAXINT;
   MINCOUR:=1;
   FOR LET:='A' TO 'Z' DO
     BEGIN
       VAL:=DIGRAM[LETAV,LET]*DIGRAM[LET,LETAP];
       IF VAL<>0
         THEN
           IF VAL <= DERVAL
           THEN
             BEGIN
               DERVAL:=VAL;
               VALREMP[MINCOUR]:=VAL;
               LETREMP[MINCOUR]:=CORSTR[LET];
               MINCOUR:=MINCOUR+1
             END
           ELSE INSERER(VAL,CORSTR[LET],VALREMP,LETREMP,MINCOUR)
     END;
   COMPTLET:=MINCOUR-1
END;

PROCEDURE REMPLACEMENT(LISTMOT:TABCAR;INDICE:TABIND;DIGRAM:MATRICE;DIMMOT,
                       DIMPOSLETCRIT:INTEGER;POSLETCRIT:TAB7;VAR ENTREE:CHAINE;
                       VAR ORTHO:BOOLEAN);
VAR IND,IND2,BIDON1,BIDON2,COMPTLET,POSCRIT:INTEGER;
    LETREMP:TAB27;
    LETAV,LETAP,LETCA:CHAR;
    TRAVAIL,REMP:CHAINE;
```

```
 PROCEDURE LETAVAP(ENTREE:CHAINE;POSCRIT:INTEGER;
                   VAR LETAV:CHAR;VAR LETAP:CHAR);
 BEGIN
      IF POSCRIT=1 THEN LETAV:='@'
                   ELSE LETAV:=ENTREE[POSCRIT-1];
      IF POSCRIT=LENGTH(ENTREE) THEN LETAP:='@'
                                ELSE LETAP:=ENTREE[POSCRIT+1]
 END;

BEGIN
  IND:=1;
  TRAVAIL:=ENTREE;
  WHILE (IND<=DIMPOSLETCRIT) AND (NOT ORTHO) DO
   BEGIN
    POSCRIT:=POSLETCRIT[IND];
    LETCA:=TRAVAIL[POSCRIT];
    LETAVAP(TRAVAIL,POSCRIT,LETAV,LETAP);
    CLASLETREMP(DIGRAM,LETAV,LETAP,LETREMP,COMPTLET);
    IND2:=1;
    WHILE (IND2<=COMPTLET) AND (NOT ORTHO) DO
     BEGIN
      REMP:=LETREMP[IND2];
      IF (REMP<>CORSTR[LETCA]) THEN
       BEGIN
         REMPLACLETTRE(POSCRIT,TRAVAIL,REMP);
         RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,BIDON1,BIDON2);
         IF (NOT ORTHO) THEN TRAVAIL:=ENTREE
                        ELSE ENTREE:=TRAVAIL
       END;
      IND2:=IND2+1
     END;
    IND:=IND+1;
   END
END;

PROCEDURE INSERTION (LISTMOT:TABCAR;INDICE:TABIND;DIGRAM:MATRICE;DIMMOT,
                     DIMFREQ,FREQNUL:INTEGER;VAR FREQUENCE:TAB7;
                     VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);
VAR IND,IND2,COMPTLET,BIDON1,BIDON2,POSCRIT:INTEGER;
    LETGAU,LETDROI:CHAR;
    INS,TRAVAIL:CHAINE;
    LETINS:TAB27;
    POSDIGCRIT:TAB7;
    DIMPOSDIGCRIT:INTEGER;

 PROCEDURE CLASPOSINS (FREQNUL,DIMFREQ:INTEGER;VAR FREQUENCE,POSDIGCRIT:TAB7;
                       VAR DIMPOSDIGCRIT:INTEGER);
 VAR IND,IND2,MIN,PM:INTEGER;
 BEGIN
  IF FREQNUL <> 0
     THEN
       BEGIN
         IF FREQNUL = 1
            THEN POSDIGCRIT[1]:=1
            ELSE POSDIGCRIT[1]:=FREQNUL;
         DIMPOSDIGCRIT:=1
       END
     ELSE
       BEGIN
         FOR IND:=1 TO DIMFREQ DO
            BEGIN
              MIN:=MAXINT;
              FOR IND2:=1 TO DIMFREQ DO
                BEGIN
                  IF FREQUENCE [IND2] < MIN THEN
                    BEGIN
                      MIN := FREQUENCE[IND2];
```

```
                    PM:=IND2
                  END
              END;
            FREQUENCE [PM]:=MAXINT;
            POSDIGCRIT[IND]:=PM;
          END;
        DIMPOSDIGCRIT:=DIMFREQ
      END
 END;

 PROCEDURE LETGAUDROI (TRAVAIL:CHAINE;POSCRIT:INTEGER;VAR LETGAU,LETDROI:CHAR);
 BEGIN
   IF POSCRIT=1 THEN LETGAU:='@'
                ELSE LETGAU:=TRAVAIL[POSCRIT-1];
   IF POSCRIT>LENGTH(TRAVAIL) THEN LETDROI:='@'
                              ELSE LETDROI:=TRAVAIL[POSCRIT]
 END;

BEGIN
 CLASPOSINS(FREQNUL,DIMFREQ,FREQUENCE,POSDIGCRIT,DIMPOSDIGCRIT);
 IND:=1;
 TRAVAIL:=ENTREE;
 WHILE(IND<=DIMPOSDIGCRIT)AND(NOT ORTHO) DO
   BEGIN
     POSCRIT:=POSDIGCRIT[IND];
     LETGAUDROI(TRAVAIL,POSCRIT,LETGAU,LETDROI);
     CLASLETREMP(DIGRAM,LETGAU,LETDROI,LETINS,COMPTLET);
     IND2:=1;
     WHILE (IND2<=COMPTLET)AND(NOT ORTHO) DO
       BEGIN
         INS:=LETINS[IND2];
         INSERT(INS,TRAVAIL,POSCRIT);
         RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,BIDON1,BIDON2);
         IF NOT ORTHO THEN TRAVAIL:=ENTREE
                      ELSE ENTREE:=TRAVAIL;
         IND2:=IND2 + 1
       END;
     IND:=IND + 1
   END
END;

PROCEDURE INTERVERTIR(LISTMOT:TABCAR;INDICE:TABIND;DIMMOT,COMMAX:INTEGER;
                      VAR ENTREE:CHAINE;VAR ORTHO:BOOLEAN);
VAR INTER:CHAR;
    IND,BIDON1,BIDON2:INTEGER;
    STRLET,TRAVAIL:CHAINE;
BEGIN
 IF LENGTH(ENTREE)>=2 THEN
  BEGIN
   IND:=1;
   TRAVAIL:=ENTREE;
   WHILE (NOT ORTHO) AND (IND<=COMMAX+1) AND (IND < LENGTH(TRAVAIL)) DO
     BEGIN
       INTER:=TRAVAIL[IND+1];
       DELETE(TRAVAIL,IND+1,1);
       INSERT(CORSTR[INTER],TRAVAIL,IND);
       RECHERCHE(LISTMOT,INDICE,DIMMOT,TRAVAIL,ORTHO,BIDON1,BIDON2);
       IF NOT ORTHO THEN TRAVAIL:=ENTREE
                    ELSE ENTREE:=TRAVAIL;
       IND:=IND+1
     END
  END
END;
```

```
PROCEDURE ESCAPE(LISTMOT:TABCAR;INDICE:TABIND;INDICETERM:TABTERM;DIMMOT,
                 DIMTERM,POSDEPART:INTEGER;VAR ENTREE:CHAINE;
                 VAR COMMAX:INTEGER;VAR ORTHO:BOOLEAN);

VAR BI,BS,LONGTERM,DEBTERMENT,DEMI,INDCAR,DISTMIN,DISTCAND:INTEGER;
    TROUVE,CONTINUE:BOOLEAN;
    TERMENT,SUFFIXE,INTER:CHAINE;

 PROCEDURE DISTANCE(ENTREE,STR:CHAINE;DISTMIN:INTEGER;VAR DIST:INTEGER);
 VAR LONG,IND,IND2:INTEGER;
     STRBIS:CHAINE;
     RESULTAT:BOOLEAN;

  PROCEDURE INTERCHANGER(VAR RESULTAT:BOOLEAN);
  BEGIN
   RESULTAT:=FALSE;
   IF ((IND2+1) <= LENGTH(STRBIS)) AND ((IND+1) <= LENGTH(ENTREE))
     THEN IF (ENTREE[IND]=STRBIS[IND2+1]) AND (ENTREE[IND+1]=STRBIS[IND2])
            THEN
             BEGIN
              DIST:=DIST+1;
              IND:=IND+2;
              IND2:=IND2+2;
              RESULTAT:=TRUE
             END
  END;

  PROCEDURE SUBSTITUER(VAR RESULTAT:BOOLEAN);
  BEGIN
   RESULTAT:=FALSE;
   IF((IND+1) <= LENGTH(ENTREE)) AND ((IND2+1) <= LENGTH(STRBIS))
     THEN IF ENTREE[(IND+1)]=STRBIS[(IND2+1)]
       THEN
         BEGIN
           DIST:=DIST+1;
           IND:=IND+2;
           IND2:=IND2+2;
           RESULTAT:=TRUE
         END
  END;

  PROCEDURE AJOUTER(VAR RESULTAT:BOOLEAN);
  BEGIN
   RESULTAT:=FALSE;
   IF (IND+1) <= LENGTH(ENTREE)
     THEN IF ENTREE[IND+1]=STRBIS[IND2]
       THEN
         BEGIN
           DIST:=DIST+1;
           LONG:=LONG+1;
           IND:=IND+2;
           IND2:=IND2+1;
           RESULTAT:=TRUE
         END
  END;

  PROCEDURE OUBLIER(VAR RESULTAT:BOOLEAN);
  BEGIN
    RESULTAT:=FALSE;
    IF (IND2+1) <= LENGTH(STRBIS)
      THEN IF ENTREE[IND]=STRBIS[IND2+1]
        THEN
          BEGIN
            DIST:=DIST+1;
            LONG:=LONG-1;
            IND:=IND+1;
            IND2:=IND2+2;
            RESULTAT:=TRUE
```

```
        END
END;

(* DEBUT PROCEDURE DISTANCE *)

BEGIN
 STRBIS:=STR;
 LONG:=LENGTH(STRBIS);
 DIST:=0;
 IND:=1;
 IND2:=1;
 WHILE (IND<=LENGTH(ENTREE)) AND (IND2<=LENGTH(STRBIS)) AND (DIST<DISTMIN) DO
  BEGIN
   IF ENTREE[IND]=STRBIS[IND2]
    THEN
     BEGIN
      IND:=IND+1;
      IND2:=IND2+1
     END
    ELSE
     BEGIN
      INTERCHANGER(RESULTAT);
      IF NOT RESULTAT
       THEN
        BEGIN
         SUBSTITUER(RESULTAT);
         IF NOT RESULTAT
          THEN
           BEGIN
            AJOUTER(RESULTAT);
            IF NOT RESULTAT
             THEN
              BEGIN
               OUBLIER(RESULTAT);
               IF NOT RESULTAT
                THEN
                 BEGIN
                  IF(IND=LENGTH(ENTREE))OR(IND2=LENGTH(STRBIS))
                       THEN DIST:=DIST+1
                       ELSE DIST:=DIST+2;
                  IND:=IND+1;
                  IND2:=IND2+1
                 END
              END
        END
     END
    END
  END;
 DIST:=DIST+ABS(LENGTH(ENTREE)-LONG)
END;

PROCEDURE LISTCANDIDAT(ENTREE:CHAINE;COMMAX,POSDEPART:INTEGER;
                       VAR DISTMIN:INTEGER;VAR INTER:CHAINE);
VAR IND2,MAXCOM,IND3,LONGUEUR:INTEGER;
    MEILLEUR,SINTER:CHAINE;
BEGIN
 MAXCOM:=COMMAX;
 IND2:=POSDEPART;
 WHILE(MAXCOM=COMMAX)AND(IND2>0)AND(DISTMIN>1) DO
   BEGIN
     IND3:=INDICE[IND2];
     LONGUEUR:=INDICE[IND2+1]-IND3-1;
     CHARBYTE(LISTMOT,IND3,LONGUEUR,SINTER);
     COMPARE(ENTREE,SINTER,MAXCOM);
     IF COMMAX=MAXCOM THEN
       BEGIN
         IND2:=IND2-1;
         DISTANCE(ENTREE,SINTER,DISTMIN,DISTCAND);
```

```
          IF DISTCAND<DISTMIN THEN
                                  BEGIN
                                    MEILLEUR:=SINTER;
                                    DISTMIN:=DISTCAND
                                  END
        END
    END;
  IND2:=POSDEPART+1;
  MAXCOM:=COMMAX;
  WHILE(MAXCOM=COMMAX)AND(IND2<316)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICE[IND2];
      LONGUEUR:=INDICE[IND2+1]-IND3-1;
      CHARBYTE(LISTMOT,IND3,LONGUEUR,SINTER);
      COMPARE(ENTREE,SINTER,MAXCOM);
      IF COMMAX=MAXCOM THEN
        BEGIN
          IND2:=IND2+1;
          DISTANCE(ENTREE,SINTER,DISTMIN,DISTCAND);
          IF DISTCAND<DISTMIN THEN
                                  BEGIN
                                    MEILLEUR:=SINTER;
                                    DISTMIN:=DISTCAND
                                  END
        END
    END;
    INTER:=MEILLEUR
  END;

PROCEDURE LISTSUFFCANDI(SUFFIXE:CHAINE;POSDEPART:INTEGER;VAR DISTMIN:INTEGER;
                        VAR INTER:CHAINE);
VAR IND2,IND3,LONGMOT,DEBTERM:INTEGER;
    MOT,SUFFCANDI:CHAINE;
    CH:CHAR;
BEGIN
  IND2:=POSDEPART;
  SUFFCANDI:=SUFFIXE;
  WHILE(SUFFCANDI=SUFFIXE)AND(IND2>0)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICETERM[IND2];
      CHARBYTE(LISTMOT,IND3,LONGTERM,SUFFCANDI);
      IF SUFFCANDI=SUFFIXE THEN
        BEGIN
          IND2:=IND2-1;
          CH:=LISTMOT[IND3];
          LONGMOT:=2;
          WHILE (CH<>'@')AND(IND3>0)DO
            BEGIN
              IND3:=IND3-1;
              LONGMOT:=LONGMOT+1;
              CH:=LISTMOT[IND3]
            END;
          IND3:=IND3+1;
          CHARBYTE(LISTMOT,IND3,LONGMOT,MOT);
          DISTANCE(ENTREE,MOT,DISTMIN,DISTCAND);
          IF DISTCAND<DISTMIN THEN
            BEGIN
              INTER:=MOT;
              DISTMIN:=DISTCAND
            END
        END
    END;
  IND2:=POSDEPART+1;
  SUFFCANDI:=SUFFIXE;
  WHILE(SUFFCANDI=SUFFIXE)AND(IND2<99)AND(DISTMIN>1) DO
    BEGIN
      IND3:=INDICETERM[IND2];
      CHARBYTE(LISTMOT,IND3,LONGTERM,SUFFCANDI);
```

```
        IF SUFFCANDI=SUFFIXE THEN
          BEGIN
            IND2:=IND2+1;
            CH:=LISTMOT[IND3];
            LONGMOT:=2;
            WHILE (CH<>'@')AND(IND3>0)DO
              BEGIN
                IND3:=IND3-1;
                LONGMOT:=LONGMOT+1;
                CH:=LISTMOT[IND3];
              END;
               IND3:=IND3+1;
               CHARBYTE(LISTMOT,IND3,LONGMOT,MOT);
               DISTANCE(ENTREE,MOT,DISTMIN,DISTCAND);
               IF DISTCAND<DISTMIN THEN
                 BEGIN
                   INTER:=MOT;
                   DISTMIN:=DISTCAND
                 END
             END
         END
    END;

(* PROCEDURE ESCAPE  *)

BEGIN
  DISTMIN:=MAXINT;
  LONGTERM:=3;
  IF COMMAX>4 THEN
                BEGIN
                   LISTCANDIDAT(ENTREE,COMMAX,POSDEPART,DISTMIN,INTER);
                   ORTHO:=TRUE;
                   ENTREE:=INTER
                END;
  IF DISTMIN=1
     THEN CONTINUE:=FALSE
     ELSE CONTINUE:=TRUE;
  BI:=1;
  BS:=DIMTERM;
  DEBTERMENT:=LENGTH(ENTREE)-2;
  TERMENT:=COPY(ENTREE,DEBTERMENT,3);
  WHILE CONTINUE AND(BI<=BS)DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICETERMI[DEMI];
      CHARBYTE(LISTMOT,INDCAR,LONGTERM,SUFFIXE);
      IF TERMENT = SUFFIXE
        THEN
          BEGIN
            CONTINUE:=FALSE;
            LISTSUFFCANDI(TERMENT,DEMI,DISTMIN,INTER);
            ORTHO:=TRUE;
            ENTREE:=INTER;
          END
        ELSE IF TERMENT < SUFFIXE
          THEN BS:=DEMI-1
          ELSE BI:=DEMI+1
    END
END;
```

```
PROCEDURE MISEPLUR(PLURIEL:BOOLEAN;VAR ENTREE:CHAINE);
VAR VOYELLES:SET OF 'A'..'U';
BEGIN
  VOYELLES:=['A','E','I','O','U'];
  IF PLURIEL THEN
   IF ENTREE[LENGTH(ENTREE)] IN VOYELLES
    THEN ENTREE:=CONCAT(ENTREE,'S')
    ELSE
     BEGIN
      IF ENTREE[LENGTH(ENTREE)]='Z'
       THEN
        BEGIN
         DELETE(ENTREE,LENGTH(ENTREE),1);
         ENTREE:=CONCAT(ENTREE,'C')
        END;
      IF NOT(ENTREE[LENGTH(ENTREE)]='S')OR NOT(LENGTH(ENTREE)>4)
          THEN ENTREE:=CONCAT(ENTREE,'ES')
     END
END;(*MISEPLUR*)
```

A N N E X E 9.

DETERMINANTS DE NOMS.

1. Articles définis.

masculin : el — masculin pluriel : los
féminin : la — féminin pluriel : las

2. Articles indéfinis.

masculin : un
féminin : una
— pluriel : pas exprimé.

3. Les démonstratifs.

a. masculin : este	pluriel : estos	} ici
féminin : esta	pluriel : estas	
b. masculin : ese	pluriel : esos	} là
féminin : esa	pluriel : esas	
c. masculin : aquel	pluriel : aquellos	} là-bas
féminin : aquella	pluriel : aquellas	

4. Les possessifs.

a. masculin et féminin : mi	pluriel : mis
b. masculin et féminin : tu	pluriel : tus
c. masculin et féminin : su	pluriel : sus
d. masculin : nuestro	pluriel : nuestros
féminin : nuestra	pluriel : nuestras
e. masculin : vuestro	pluriel : vuestros
féminin : vuestra	pluriel : vuestras

A N N E X E 10.

CLASSIFICATION DES MOTS DU VOCABULAIRE EN CATEGORIES.

		+humain	+animé	+commun	+propre	+abstrait	+concret	+collectif	+individu	+nombrable	+masculin	+mâle
agua		-	-	+	-	-	+	-	+	-	-	-
aire		-	-	+	-	-	+	-	+	-	+	+
almacén		-	-	+	-	-	+	-	+	+	+	+
amigo		±	+	+	-	-	+			+	+	+
ano		-	-	+	-	+	-			+	+	+
aparador		-	-	+	-	-	+			+	+	+
apendicitis		-	-	+	-	-	+	-		+	-	-
árbol		-	-	+	-	-	+			+	+	+
armario		-	-	+	-	-	+			+	+	+
artista		+	+	+	-	-	+			+	±	±
asesino		+	+	+	-	-	+			+	±	±
asesinato		-	-	+	-	-	+			+	+	+
ataque		-	-	+	-	-	+			+	+	+
autobús		-	-	+	-	-	+			+	+	+
avenida		-	-	+	-	-	+			+	-	-
avería	1.	-	-	+	-	-	+			+		
	2.	-	-	+	-	-	+			+	-	-
azulejo		-	+	+	-	-	+			+	+	+
balcón		-	-	+	-	-	+			+	+	+
barrio		-	-	+	-	-	+			+	+	+
beneficensia		-	-	+	-	+	-			-	-	-
botón		-	-	+	-	-	+			+	+	+
brasero		-	-	+	-	-	+			+	+	+
brazo		-	-	+	-	-	+			+	+	+
brocha		-	-	+	-	-	+			+	-	-
burro		-	+	+	-	-	+			+	+	+
cafetera		+	+	+	-	-	+			+	-	-
caja		-	-	+	-	-	+			+	-	-
calle		-	-	+	-	-	+			+	-	-
calor		-	-	+	-	±	±			±	+	+
cama		-	-	+	-	-	+	-	+	+	-	-
camina		-	-	+	-	-	+			+	+	+

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
camisa	-	-	+	-	-	+	-	+	+	-	-
campo	-	-	+	-	-	+			+	+	+
cantidad	-	-	+	-	+	-				-	-
cara	-	-	+	-	-	+	-	+	+	-	-
carbón	-	-	+	-	-	+			+	+	+
cárcel	-	-	+	-	-	+			+	-	-
carne	±	-	+	-	-	+				-	-
carta	-	-	+	-	-	+	-	+	+	-	-
cartel	-	-	+	-	-	+	-	+	+	+	+
casa	-	-	+	-	-	+			+	-	-
cita	-	-	+	-	+	-	-	+	+	-	-
cena	-	-	+	-	-	+	-	+	+	-	-
cerveza	-	-	+	-	-	+	-	+	+	-	-
churro	-	-	+	-	-	+	-	+	+	+	+
ciego	±	±	+	-	-	+	-	+	+	±	±
cine	-	-	+	-	-	+				+	+
cinturón	-	-	+	-	-	+			+	+	+
ciudad	-	-	+	-	-	+				-	-
cobre	-	-	+	-	-	+			+	+	+
coche	-	-	+	-	-	+			+	+	+
cocido	-	-	+	-	-	+			+	+	+
cocina	-	-	+	-	-	+			+	-	-
col	-	-	+	-	-	+			+	-	-
cola	-	-	+	-	-	+	-	+	+	-	-
color	-	-	+	-	-	+			+	+	+
comedor	-	-	+	-	-	+			+	+	+
comisario	+	+	+	-	-	+	-	+	+	+	+
cómoda	-	-	+	-	-	+			+	-	-
compania	-	-	+	-	-	+	+	-	+	-	-
corazón	±	-	+	-	±	±	-	+	+	+	+
cosa	-	-	+	-	-	+			+	-	-
costa	-	-	+	-	-	+			+	-	-
cuarto	-	-	+	-	-	+			+	+	+
cuchillo	-	-	+	-	-	+	-	+	+	+	+
cuerda	-	-	+	-	-	+	-	+	+	-	-
cuero	-	-	+	-	-	+	-	+	+	+	+
curso	-	-	+	-	-	+			+	+	+

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
dano	-	-	+	-	+	-				+	+
décimo	-	-	+	-	-	+	-	+	+	+	+
dentista	+	+	+	-	-	+	-	+	+	±	±
despertador	-	-	+	-	-	+	-	+	+	+	+
desván	-	-	+	-	-	+	-	+	+	+	+
día	-	-	+	-	-	+	-	+	+	+	+
diente		-	+	-	-	+	-	+	+	+	+
dinero	-	-	+	-	-	+			+	+	+
dirección	-	-	+	-	-	+			+	-	-
disco	-	-	+	-	-	+	-	+	+	+	+
discussión	-	-	+	-					+	-	-
doctor	+	+	+	-	-	+	-	+	+	±	±
dueno	+	+	+	-	-	+	-	+	+	+	+
edificio	-	-	+	-	-	+			+	+	+
efecto	-	-	+	-	-	+			+	+	+
ejército			+	-	-	+	+	-	+	+	+
empleado	+	+	+	-	-	+	-	+	+	±	±
enfermedad	-	-	+	-						-	-
ensalada	-	-	+	-	-	+	-	+	+	-	-
escayola	-	-	+	-	-	+	-	+	+	-	-
escopeta	-	-	+	-	-	+	-	+	+	-	-
espada	-	-	+	-	-	+	-	+	+	-	-
espalda	±	-	+	-	-	+	-	+	+	-	-
esposo	+	+	+	-	-	+	-	+	+	+	+
esposa	+	+	+	-	-	+	-	+	+	-	-
esquina	-	-	+	-	-	+			+	-	-
estación	-	-	+	-	-	+			+	-	-
estatua	-	-	+	-	-	+			+	-	-
estilo	-	-	+	-	+	-			+	+	+
éxito	-	-	+	-	+	-			+	+	+
fábrica	-	-	+	-	-	+			+	-	-
faena	-	-	+	-				+	+	-	-
falda	-	-	+	-	-	+	-	+	+	-	-
familia	-	±	+	-	±		+	-		-	-
fiebre	-	-	+	-		+				-	-
flor	-	-	+	-	-	+	-	+	+	-	-
forma	-	-	+	-						-	-

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
foto	-	-	+	-	-	+	-	+	+	-	-
fresa	-	-	+	-	-	+	-	+	+	-	-
fresón	-	-	+	-	-	+	-	+	+	+	+
frío	-	-	+	-		+			+	+	+
fruta	-	-	+	-	-	+	-	+	+	-	-
fuente	-	-	+	-	-	+	-	+	+	-	-
fútbol	-	-	+	-	-	+			+	+	+
gafas	-	-	+	-	-	+	-	+	+	-	-
garage	-	-	+	-	-	+	-	+	+	+	+
garbanzo	-	-	+	-	-	+	-	+	+	+	+
gato	-	+	+	-	-	+	-	+	+	+	+
gente	+	+	+	-	-	+	+	-	+	-	-
gitano	+	+	+	-	-	+	-	+	+	+	+
guitarra	-	-	+	-	-	+	-	+	+	-	-
habitación	-	-	+	-	-	+			+	-	
hallazgo	-	-	+	-						+	
hombre	+	+	+	-	-	+	-	+	+	+	+
hombros	±	-	+	-	-	+	-	+	+	+	
hora	-	-	+	-						-	
hospital	-	-	+	-	-	+				+	
huésped	+	+	+	-	-	+	-	+	+	+	
huevo	-	-	+	-	-	+	-	+	+	+	
iglesia	-	-	+	-	-	+				-	
instrumento	-	-	+	-	-	+	-	+	+	+	
invierno	-	-	+	-	-	+				+	
jaleo		-	+	-	-	+	-	+	+	+	
jamón	-	-	+	-	-	+			+	+	
jarro	-	-	+	-	-	+	-	+	+	+	
jefe	+	+	+	-	-	+	-	+	+	+	+
jota		-	+	-	-	+	-		-	-	
judías	-	-	+	-	-	+	-	+	+	-	
kilo	-	-	+	-	-	+	-	+	+		
kilómetro	-	-	+	-	-	+	-	+	+		
labor	-	-	+	-							
lana	-	-	+	-	-	+			+		
lástima	-	-	+	-	+	-					
lena	-	-	+	-	-	+			+		

		±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
libro		-	-	+	-	-	+	-	+	+		
lluvia		-	-	+	-	-	+					
loco			-	+	-							
lugar		-	-	+	-	-	+	-	+	+		
luz		-	-	+	-	-	+					
maceta		-	-	+	-	-	+	-	+	+		
madeja		±	-	+	-	-	+	-	+	+		
maleta		-	-	+	-	-	+	-	+	+		
mano				+	-	-	+	-	+	+		
mantilla		-	-	+	-	-	+	-	+	+		
manzana		-	-	+	-	-	+	-	+	+		
manana		-	-	+	-	-	+	-	+	+		
mapa	1.	+	+	+	-	-	+	-	+	+		
	2.	-	-	+	-	-	+	-	+	+		
máquina		-	-	+	-	-	+	-	+	+		
mar		-	-	+	-	-	+					
maravilla		-	-	+	-							
mecánico		+	+	+	-	-	+	-	+	+		
médico		+	+	+	-	-	+	-	+	+		
medicina		-	-	+	-	-	+	-	+	+		
mercado		-	-	+	-	-	+					
merluza		-	-	+	-	-	+	-	+			
mes		-	-	+	-	+	-					
mesa		-	-	+	-	-	+			+		
mesón		-	-	+	-	-	+			+		
metro		-	-	+	-	-	+			+		
mudo				+	-							
minuto		-	-	+	-	+	-			+		
molino	1.	-	-	+	-	-	+					
	2.	+	+	+	-	-	+					
montana		-	-	+	-	-	+			+		
monte		-	-	+	-	-	+			+		
música		-	-	+	-	+	-			-		
mujer		+	+	+	-	-	+			+		

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
naranja	-	-	+	-	-	+	-	+	+		
nariz	-	-	+	-	-	+			+		
negro	+	+	+	-	-	+			+		
nieto	+	+	+	-	-	+	-	+	+		
nino	+	+	+	-	-	+	-	+	+		
noche	-	-	+	-	+	-					
nombre	-	-	+	-	+	-					
novio	+	+	+	-	-	+			+		
número	-	-	+	-	+	-			+		
obra	-	-	+	-	-	+			+		
ocasión	-	-	+	-	-	+			+		
ojo	-	-	+	-	-	+	-	+	+		
oreja	-	-	+	-	-	+	-	+	+		
orquesta	-	-	+	-	-	+	+	-	+		
padre	+	+	+	-	-	+	-	+	+	+	+
paella	-	-	+	-	-	+	-	+	+		
página	-	-	+	-	-	+	-	+	+		
paisaje	-	-	+	-	-	+	+	-	+		
pájaro	-	+	+	-	-	+	-	+	+		
palacio	-	-	+	-	-	+			+		
pan	-	-	+	-	-	+			+		
par	-	-	+	-			+	-	+		
paragüero	-	-	+	-	-	+	+	-	+		
pared	-	-	+	-	-	+			+		
parque	-	-	+	-	-	+	+	-	+		
parte	-	-	+	-					+		
partido	-	-	+	-	-	+			+		
paseo	-	-	+	-					+		
patata	-	-	+	-	-	+	-	+	+		
patio	-	-	+	-	-	+			+		
peineta	-	-	+	-	-	+	-	+	+		
película	-	-	+	-	-	+			+		
pelo	-	-	+	-	-	+			+		
periódico	-	-	+	-	-	+	-	+	+		
perro	-	+	+	-	-	+	-	+	+		
pescadilla	-	-	+	-	+	-	-	+	+		

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
pesca	-	-	+	-							
pescado	-	+	+	-	-	+	-	+	+		
peseta	-	-	+	-	-	+			+		
pez	-	+	+	-	-	+	-	+	+		
pie	-	-	+	-	-	+	-	+	+		
piedra	-	-	+	-	-	+	-	+	+		
pieza	-	-	+	-	-	+			+		
pintor	+	+	+	-	-	+	-	+	+		
piso	-	-	+	-	-	+			+		
playa	-	-	+	-	-	+					
plaza	-	-	+	-	-	+					
policia	±		+	-	-	+					
portal	-	-	+	-	-	+			+		
portera	+	+	+	-	-	+	-	+	+		
prácticás	-	-	+	-	-	+			+		
precio	-	-	+	-	+	-			+		
primo	+	+	+	-	-	+	-	+	+		
prisa	-	-	+	-	+	-					
problema	-	-	+	-	+	-			+		
profesión	-	-	+	-	+	-					
provincia	-	-	+	-	-	+			+		
pueblo	-	-	+	-	-	+			+		
puente	-	-	+	-	-	+			+		
puerta	-	-	+	-	-	+			+		
puerto	-	-	+	-	-	+			+		
pulsera	-	-	+	-	-	+	-	+	+		
puro	-	-	+	-	-	+	-	+	+		
rato	-	-	+	-	+	-			+		
ratón	-	+	+	-	-	+	-	+	+		
recibidor	-	-	+	-	-	+			+		
recipiente	-	-	+	-	-	+	-	+	+		
región	-	-	+	-	-	+			+		
reloj	-	-	+	-	-	+	-	+	+		
restaurante	-	-	+	-	-	+			+		
rico	+	+	+	-	-	+	-	+	+		
rincón	-	-	+	-	-	+			+		

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
río	-	-	+	-	-	+			+		
romance	-	-	+	-	+	-			+		
ropa	-	-	+	-	-	+	-	+	+		
rubio	-	-	+	-							
ruido	-	-	+	-	-	+			+		
sala	-	-	+	-	-	+			+		
salud	-	-	+	-	+	-				-	
sangría	-	-	+	-	-	+				-	
santo	+	+	+	-	-	+			+	+	-
santa	+	+	+	-	-	+			+	-	+
sed	-	-	+	-	+	-				-	
sello	-	-	+	-	-	+	-	+	+	+	
semana	-	-	+	-	+	-			+	-	
senor	+	+	+	-	-	+	-	+	+	+	
sereno	+	+	+	-	-	+	-	+	+	+	
sierra	-	-	+	-	-	+			+	-	
silla	-	-	+	-	-	+	-	+	+	-	
sitio	-	-	+	-	-	+			+	+	
situación	-	-	+	-	+	-			+		
sobrino	+	+	+	-	-	+	-	+	+		
sobrina	+	+	+	-	-	+	-	+	+		
sol	-	-	+	-	-	+					
sombrero	-	-	+	-	-	+	-	+	+		
sótano	-	-	+	-	-	+			+		
suelo	-	-	+	-	-	+			+		
sueno	-	-	+	-	+	-	-	+	+		
suerte	-	-	+	-	+	-			+		
susto	-	-	+	-	+	-			+		
tacón	-	-	+	-	-	+	-	+	+		
tabler	-	-	+	-	-	+			+		
tapa	-	-	+	-	-	+			+		
tapete	-	-	+	-	-	+	-	+	+		
tarde	-	-	+	-	+	-			+		
tasca	-	-	+	-	-	+	+	-	+		
teatro	-	-	+	-	-	+	+	-	+		
televisión	-	-	+	-	-	+	-	+	+		

	±humain	±animé	±commun	±propre	±abstrait	±concret	±collectif	±individuel	±nombrable	±masculin	±mâle
temporado	-	-	+	-	-	+					
terraza	-	-	+	-	-	+			+		
tiempo	-	-	+	-	+	-					
tiopa	+	+	+	-	-	+	-	+	+		
tiro	-	-	+	-	-	+	-	+	+		
tocadiscos	-	-	+	-	-	+	+		+		
tomate	-	-	+	-	-	+	-	+	+		
tómbola	-	-	+	-	-	+	+	-	+		
tanto	+	+	+	-	-	+	-	+	+		
torero	+	+	+	-	-	+	-	+	+		
toro	-	+	+	-	-	+	-	+	+		
tortilla	-	-	+	-	-	+	-	+	+		
trabajo	-	-	+	-							
traje	-	-	+	-	-	+	-	+	+		
tren	-	-	+	-	-	+	+	-	+		
vaca	-	+	+	-	-	+	-	+	+		
vacaciones	-	-	+	-	+	-					
vecino	+	+	+	-	-	+	-	+	+		
ventana	-	-	+	-	-	+			+		
verano	-	-	+	-	+	-					
verdad	-	-	+	-	+	-					
verdura	-	-	+	-	-	+			+		
vez	-	-	+	-					+		
viaje	-	-	+	-	-	+			+		
viejo	+	+	+	-	-	+	-	+	+		
vino	-	-	+	-	-	+			+		
zapato	-	-	+	-	-	+	-	+	+		
zarzuela	-	-	+	-					+		
zona	-	-	+	-	+	-			+		
zumo	-	-	+	-	-	+			+		

	±transitif	±attributif	±avoir	±actif	±duratif	±perfectif	±sujet = humain	±objet animé	± A	± EN
estar	-	-	+	+	+	-	+	-/+	-	+
pasar	-	+	+	+	+	-	+	-	-	+
vivir	+	+	+	+	+	-	+	-	-	+
situar(se)	-	+	+	+	+	-	+	-/+	-	+
comer	+	+	+	+	+	-	+	-/+	-	+
quedar(se)	-	+	+	+	+	-	+	-	-	+
morir	-		+	+	+	-	+		-	+
ir	-		+	+	+	-	+	-	+	-
llegar	+		+	+	+	-	+	-	+	+
marchar(se)	+		+	+	+	-	+	-	+	-
subir	+		+	+	+	-	+	-	+	-
venir	-		+	+	+	-	+	-	+	-
nadar	-		+	+	+	-	+	-	+	+
trabajar	+		+	+	+	-	+	-	-	+
meter(se)	+		+	+	+	-	+	-	-	+

A N N E X E 11.

LISTES DE TOUS LES SUJETS DE CHAQUE VERBE.

1. Sujets de vivir.

amigo - arbol - artista - asesino - azulejo - burro - ciego - comisario - corazon - dentista - doctor - dueno - ejercito - empleado - ensalada - esposa - esposo - familia - flor - gato - gente - gitano - hombre - huesped - jefe - loco - mecanico - medico - mudo - mujer - negro - nieto - nino - novio - orquesta - padre - pajaro - perro - pintor - policia - portera - primo - raton - rico - santa - santo - senor - sereno - sobrino - sobrina - tio/tia - tonta - tonto - torero - toro - vaca - vecino - viejo -
+ les prénoms et les noms
+ nationalité
+ les pronoms sujets

2. Sujets de trabajar.

amigo - artista - asesino - burro - ciego - comisario - dentista - doctor - dueno - ejercito - empleado - esposa - esposo - familia - gato - gente - gitano - hombre - huesped - jefe - loco - mecanico - medico - mudo - mujer - negro - nieto - nino - novio - orquesta - padre - pescado - pez - pintor - policia - portera - primo - rico - senor - sereno - sobrina - sobrino - tio/tia - tonta - tonto - torero - vecino - viejo -
+ les prénoms et les noms
+ nationalités
+ les pronoms sujets

3. Sujets de meter(se).

agua - amigo - artista - asesino - azulejo - burro - calor - ciego - comisario - dano - dentista - doctor - dueno - ejercito - empleado - esposa - esposo - familia - freo - gato - gente - gitano - hombre - huesped - jefe - loco - mano - mar - mecanico - medico - mudo - mujer - negro - nieto - nino - novio - padre - pajaro - perro - pescado - pez - pintor - policia - portera -

primo - raton - rico - rio - senor - sereno - sobrina - sobrino - tio/tia - tonta - tonto - torero - toro - vaca - vecino - viejo -
+ les prénoms et noms
+ nationalités
+ les pronoms sujets

4. Sujets de comer.

amigo - artista - asesino - azulejo - burro - ciego - comisario - dentista - doctor - dueno - ejercito - empleado - esposa - esposo - familia - gato - gente - gitano - hombre - huesped - jefe - loco - mecanico - medico - mudo - mujer - negro - nieto - nino - novio - orquesta - padre - pajaro - perro - pescado - pez - pintor - policia - portera - primo - raton - rico - senor - sereno - sobrina - sobrino - tio/tia - tonta - tonto - torero - toro - vaca - vecino - viejo -
+ les prénoms et noms
+ nationalités
+ les pronoms sujets

5. Sujets de morir.

amigo - arbol - artista - asesino - azulejo - boton - burro - ciego - comisario - dentista - doctor - dueno - ejercito - empleado - ensalada - esposa - esposo - familia - gato - gente - gitano - hombre - huesped - jefe - loco - mecanico - medico - mudo - mujer - negro - nieto - nino - novio - padre - pajaro - perro - pescado - pez - pintor - policia - portera - primo - raton - rico - senor - sereno - sobrina - sobrino - tio/tia - tonta - tonto - torero - toro - vaca - vecino - viejo -
+ les prénoms et noms
+ nationalités
+ les pronoms sujets

6. Sujets de pasar.

aire - amigo - artista - asesino - autobus - azulejo - burro - camino - ciego - coche - comisario - dentista - doctor - dueno - ejercito - empleado - esposa - esposo - exito - familia - gato - gente - gitano - hombre - huesped - jefe - loco - mano - maquina - mar - mecanico - medico - mudo - mujer - negro - nieto - nino - novio - orquesta - padre - pajaro - perro - pintor - policia - portera - primo - raton - rico - rio - senor - sereno - sobrina - sobrino - sol - tio/tia - tonta - tonto - torero - toro - tren - vaca - vecino - viejo -
+ les prénoms et noms
+ nationalités
+ les pronoms sujets.

7. Sujets de marchar(se).

artista - asesino - azulejo - burro - ciego - coche - comisario - dentista - doctor - dueno - ejercito - empleado - esposa - esposo - familia - gato - gente - gitano - hombre - huesped - jefe - loco - mecanico - medico - modo - mujer - negro - nieto - nino - novio - orquesta - padre - pajaro - perro - pescado - pez - piedra - pintor - policia - portera - primo - raton - rico - senor - sereno - sobrina - sobrino - tio/tia - tonta - tonto - torero - toro - tren - vaca - vecino - viejo -
+ les prénoms et les noms
+ nationalités
+ les pronoms sujets

8. Sujets de quedar(se).

agua - aire - amigo - artista - asesino - azulejo - burro - carta- cerveza - churro - ciego - cobre - coche - cocido - col - cola - comisario - comoda - cuchillo - cuerda - dano - decimo - dentista - desperador - diente - dinero - direccion - disco - doctor - dueno - efecto - ejercito- empleado - ensalada - esposa - esposo - estacion - estatua - exito - familia - flor - foto - gato - gente - gitano - guitarra - hombre - hospital - huesped - huevo - instrumente - jamon - jefe - judias - lena - libro - loco - maceta - madeja - maleta - mano - mantilla -

manzana - maquina - mar - mecanico - medico - modo - mujer -
musica - naranja - nariz - negro - nieto - nino - novio -
obra - orquesta - padre - pajaro - perro - pescado - peseta -
pez - pintor - policia - portera - primo - raton - rico - senor -
sereno - sobrina - sobrino - tio/tia - tonta - tonto - torero -
toro - tren - vaca - vecino - viejo -
+ les prénoms et noms
+ nationalités
+ les pronoms sujets

9. Sujets de ir.

agua - aire - amigo - artista - asesino - autobus - azulejo -
burro - camino - ciego - coche - comisario - dentista - doctor -
dueno - ejercito - empleado - esposa - esposo - exito - familia -
gato - gente - gitano - hombre - huesped - jefe - loco - mano -
maquina - mar - mecanico - medico - mudo - mujer - negro -
nieto - nino - novio - orquesta - padre - pajaro - perro -
pescado - pez - pintor - policia - portera - primo - pulsera -
puro - raton - rico - rio - senor - sereno - sobrina - sobrino -
sol - tio/tia - tonta - tonto - torero - toro - tren - vaca -
vecino - viejo -
+ les prénoms et noms
+ nationalités
+ les pronoms sujets

10. Sujets de llegar.

agua - aire - amigo - artista - asesino - autobus - azulejo -
burro - camino - carta - ciego - coche - comisario - dano -
dentista - doctor - dueno - ejercito - empleado - esposa - esposo -
exito - familia - gato - gente - gitano - hombre - huesped -
jaleo - jefe - loco - maquina - mar - mecanico - medico - mudo -
mujer - negro - nieto - nino - novio - orquesta - padre -
pajaro - perro - pescado - pez - piedra - pintor - policia -
portera - primo - raton - rico - rio - senor - sereno - sobrina -
sobrino - sol - tio/tia - tonta - tonto - torero - toro - tren -
vaca - vecino - viejo -
+ les noms et prénoms
+ nationalités
+ les pronoms sujets

11. Sujets de subir.

agua - aire - amigo - artista - asesino - azulejo - burro -
calor - camino - ciego - comisario - cuerda - cuero - dano -
dentista - doctor - dueno - ejercito - empleado - ensalada -
esposa - esposo - exito - familia - flor - freo - gato - gente -
gitano - hombre - huesped - jaleo - jefe - loco - mano -
maquina - mar - mecanico - medico - mudo - mujer - negro -
nieto - nino - novio - orquesta - padre - pajaro - perro -
pescado - pez - pintor - policia - portera - primo - raton -
rico - senor - sereno - sobrina - sobrino - sol - tio/tia -
tonta - tonto - torero - toro - tren - vaca - vecino - viejo -
+ les noms et prénoms
+ nationalités
+ les pronoms sujets

12. Sujets de nadar.

amigo - artista - asesino - azulejo - burro - ciego - comisario -
dentista - doctor - dueno - empleado - esposa - esposo - gente -
gitano - hombre - huesped - jefe - loco - mecanico - medico -
mudo -mujer - negro - nieto - nino - novio - padre - perro -
pescado - pez - pintor - portera - primo - rico - senor -
sereno - sobrina - sobrino - tio/tia - tonta - tonto -
vecino - viejo -
+ les noms et prénoms
+ nationalités
+ les pronoms sujets

13. Sujets de venir.

aire - almacen - amigo - artista - asesino - autobus - avenida -
averia - azulejo - burro - camisa - coche - comisario - compania -
dentista - doctor - dueno - efecto - ejercito - empleado -
esposa - esposo - exito - fabrica - familia - gato - gente -
gitano - hombre - huesped - jefe -lluvia - loco - maquina -
mar - mecanico - medico - mujer - negro - nieto - nino - novio -
orquesta - padre - pajaro - perro - pescado - pez - pintor -
policia - portera - primo - pueblo - raton - rico - rio -

ruido - santa - santo - sed - senor - sereno - sobrina -
sobrino - sol - tio/tia - torero - toro - tren - vaca - vecino -
viejo -
+ les prénoms et les noms
+ nationalités
+ les pronoms sujets

14. Sujets de estar.

agua - aire - almacen - amigo - aparador - arbol - armario -
artista - asesino - autobus - avenida - azulejo - balcon -
barrio - brasero - brocha - burro - cafetera -caja - calle -
cama - camisa - campo - cara - carbon - carcel - carne - carta -
cartel - casa - cerveza - churro - ciego - cine - cinturon -
cita - ciudad - cobre - coche - cocido - cocina - col - cola -
comedor - comisario - comoda - corazon - cuarto - cuchillo -
cuerda - cuero - curso - dano - decimo - dentista - desperador -
desvan - diente - dinero - direccion - disco - doctor - dueno -
edificio - efecto - ejercito - empleado - ensalada - escopeta -
espada - espalda - esposa - esposo - estacion - estatua -
exito - fabrica - falda - familia - flor - foto - fresa -
freson - frio - fruta - fuente - gafas - garage - garbanzo -
gato - gente - gitano - guitarra - habitation - hombre -
hospital - huesped - huevo - iglesia instrumento - jamon -
jarro - jefe - judias - lena - libro - lluvia - loco - maceta -
madeja - maleta - mano - mantilla - manzana - mapa - maquina -
mar - maravilla - mecanico - medico - mercado - mesa - meson -
metro - molino - montana - monte - mudo - mujer - musica -
naranja - nariz - negro - nieto - nino - novio - obra - ojo -
oreja - orquesta - padre - paella - pagina - pajaro - palacio -
pan - paraguero - pared - parque - partido - paseo - patata -
patio - peineta -pelo - periodico - perro - pescado - peseta -
pez - pie - piedra - pieza - pintor - piso - playa - plaza -
policia - portal - portera - primo - provincia - pueblo -
puente - puerta - puerto - pulsera - puro - raton - recibidor -
recipiente - region - reloj - restaurante - rico - ropa -
ruido - sala - senor - sereno - sierra - silla - sobrina -
sobrino - sol - sombrero - sotano - tacon - taller - tapa -
tapete - tarde - tasca - teatro - television - temporada -
terraza - tiempo - tio/tia - tocadiscos - tomate - tonta -
tonto - torero - toro - tortilla - traje - tren - vaca -

vecino - verdura - viejo - vino - zapato - zarzuela -

15. Sujets de situar(se).

agua - aire - almacen -amigo - aparador - arbol - armario -
artista - asesino - autobus - avenida - averia - azulejo -
balcon - barrio - boton - brasero - brocha - burro - cafetera -
caja - calle - calor - cama - camisa - campo - cara - carbon -
carcel - carne - carta - cartel - casa - cerveza - churro -
ciego - cine - cinturon - cita - ciudad - cobre - coche -
cocido - cocina - col - cola - comedor - comisario - corazon -
cuarto - cuerda - cuero - curso - dano - decimo - dentista -
despertador - desvan - diente - dinero - direccion - disco -
doctor - dueno - edificio - efecto - ejercito - empleado -
ensalada - escopeta - espada - espalda - esposa - esposo -
estacion - estatua - exito - fabrica - falda - familia -
flor - foto - fresa - freson - freo - fruta - fuente - gafas -
garage - garbanzo - gato - gente - gitano - guitarra -
habitation - hombre - hospital - huesped - huevo - iglesia -
instrumento - jamon - jarro - jefe - judias - lena - libro -
loco - maceta - mano - mantilla - manzana - mapa - maquina -
mar - maravilla - mecanico - medico - mercado - mesa - meson -
metro - molino - montana - monte - mudo - mujer - musica -
naranja - nariz - negro - nieto - nino - novio - obra - ojo -
oreja - orquesta - padre - paella - pagina - pajaro - palacio -
pan - paraguero - pared - parque - partido - paseo - patata -
patio - peineta - pelo - periodico - perro - pescado - peseta -
pez - pie - piedra - pieza - pintor - piso - playa - plaza -
policia - portal - portera - primo - provincia - pueblo -
puente - puerta - puerto - pulsera - purro - raton - recibidor -
recipiente - region - reloj - restaurante - rico - rincon - rio -
ruido - sala - senor - sereno - sierra - silla - sobrina -
sobrino - sol - sombrero - sotano - tacon - taller - tapa -
tapete - tarde - tasca - teatro - television - temporada -
terraza - tiempo - tio/tia - tocadiscos - tomate - tonta -
tonto - torero - toro - tren - vaca - vecino - viejo - vino -
zapato - zarzuela -

A N N E X E 12.

LISTE DE TOUS LES LIEUX DE CHAQUE VERBE.

1. Lieux de vivir.

agua - aire - almacen - aparador - arbol - armario - autobus - avenida - averia - balcon - barrio - cama - camisa - campo - carcel - casa - cine - ciudad - cocina - comedor - costa - cuarto - desvan - edificio - ejercito - esquina - estacion - exito - fabrica - familia - fruta - fuente - garage - habitacion - hospital - iglesia - labor - mar - mercado - meson - molino - montana - monte - palacio - parque - patio - piso - playa - plaza - provincia - pueblo - puerto - region - restaurante - rio - ropa - ruido - sala - sierra - sitio - sotano - tasca - teatro - terraza - tren - zona -
+ les noms propres de lieu.

2. Lieux de trabajar.

agua - aire - almacen - aparador - arbol - armario - autobus - avenida - averia - balcon - barrio - cama - camisa - campo - carbon - carcel - casa - cine - ciudad - cocina - comedor - costa - cuarto - curso - desvan - edificio - ejercito - esquina - estacion - exito - fabrica - familia - fuente - garage - habitacion - hospital - iglesia - labor - lluvia - mar - mercado - meson - molino - montana - monte - orquesta - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerta - puerto - recibidor - region - restaurante - rincon - rio - ropa - ruido - sala - semana - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - zona -
+ les noms propres de lieu.

3. Lieux de meter(se).

agua - aire - almacen - aparador - armario - autobus - avenida - averia - barrio - cama - camisa - campo - carcel - casa - cine - ciudad - cocina - comedor - costa - cuarto - curso - desvan - edificio - ejercito - estacion - exito - fabrica - familia -

fruta - garage - gato - habitacion - hospital - iglesia - mar - meson - molino - ojo - oreja - orquesta - paisaje - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerta - puerto - recibidor - region - restaurante - rincon - rio - ropa - sala - sitio - sotano - taller - tasca - teatro - terraza - tren - zapato - zona -
+ les noms propres de lieu.

4. Lieux de comer.

agua - aire - almacen - autobus - avenida - averia - barrio - cama - camisa - campo - carcel - casa - cine - ciudad - cocina - comedor - cuarto - curso - desvan - edificio - ejercito - esquina - estacion - fabrica - familia - fuente - garage - gato - habitacion - hospital - labor - lluvia - mar - meson - molino - montana - monte - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerto - recibidor - recipiente - region - restaurante - rincon - rio - ropa - ruido - sala - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - zona -
+ les noms propres de lieu.

5. Lieux de morir.

agua - aire - almacen - autobus - avenida - averia - barrio - cama - camisa - campo - carcel - casa - cine - ciudad - cocina - comedor - costa - cuarto - curso - desvan - edificio - ejercito - esquina - estacion - exito - fabrica - familia - fuente - garage - habitacion - hospital - iglesia - labor - lluvia - mar - meson - molino - montana - monte - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerto - recibidor - region - restaurante - rincon - rio - ropa - ruido - sala - semana - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - zona -
+ les noms propres de lieu.

6. Lieux de pasar.

agua - aire - almacen - autobus - avenida - averia - barrio - camino - camisa - campo - casa - cine - ciudad - cocina - comedor - costa - cuarto - desvan - edificio - ejercito - estacion - exito - fabrica - familia - fuente - garage - gato - habitacion - hospital - labor - lluvia - mar - meson - molino - montana - monte - ojo - oreja - orquesta - paisaje - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerta - puerto - recibidor - region - restaurante - rincon - rio - ropa - ruido - sala - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - zona -
+ les noms propres de lieu.

7. Lieux de marchar(se).

agua - almacen - amigo - aparador - arbol - armario - artista - asesino - autobus - avenida - averia - balcon - barrio - brasero - brazo - cama - camino - camisa - campo - carbon - carcel - cartel - casa - ciego - cine - cita - ciudad - cocina - comedor - comoda - costa - cuarto - curso - dentista - despertador - desvan - doctor - dueno - edificio - ejercito - empleado - esposa - esposo - esquina - estacion - estatua - exito - fabrica - familia - flor - fruta - fuente - garage - gato - gente - gitano - habitacion - hombre - hospital - huesped - huevo - iglesia - jefe - labor - maceta - manzana - maquina - mar - mecanico - mercado - mesa - meson - molino - montana - monte - mujer - negro - nieto - nino - novio - orquesta - padre - pajaro - palacio - pared - parque - paseo - patio - perro - pescado - pez - piedra - pintor - piso - playa - plaza - policia - portal - portera - primo - provincia - pueblo - puente - puerta - puerto - raton - recibidor - region - restaurante - rico - rincon - rio - ruido - sala - senor - sereno - sierra - silla - sitio - sobrina - sobrino - sol - sotano - taller - tasca - teatro - television - terraza - tio/tia - tocadiscos - torero - toro - tren - vaca - vecino - ventana - viejo - zona -
+ les noms propres de lieu.

8. Lieux de quedar(se).

agua - aire - almacen - aparador - armario - autobus - avenida - averia - barrio - cama - camino - camisa - campo - carbon - carcel - cine - cita - ciudad - cocina - comedor - comoda - costa - cuarto - curso - desvan - edificio - ejercito - ensalada - esquina - estacion - exito - fabrica - familia - fruta - fuente - garage - habitacion - hospital - iglesia - labor - lluvia - maceta - mar - mercado - mesa - meson - molino - montana - monte - ojo - oreja - orquesta - paisaje - palacio - parque - piso - playa - plaza - portal - provincia - pueblo - puerta - puerto - recibidor - region - restaurante - rincon - rio - ropa - ruido - sala - semana - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - zapato - zona -
+ les noms propres de lieu.

9. Lieux de ir.

agua - almacen - amigo - aparador - arbol - armario - artista - asesino - autobus - avenida - averia - balcon - barrio - brasero - brazo - cama - camino - campo - carbon - carcel - cartel - casa - ciego - cine - cita - ciudad - cocina - comedor - comoda - costa - cuarto - curso - dentista - despertador - desvan - doctor - dueno - edificio - ejercito - empleado - esposa - esposo - esquina - estacion - estatua - exito - fabrica - familia - flor - fruta - fuente - garage - gato - gente - gitano - habitacion - hombre - hospital - huesped - huevo - iglesia - jefe - labor - maceta - manzana - maquina - mar - mecanico - medico - mercado - mesa - meson - molino - montana - monte - mujer - negro - nieto - nino - novio - orquesta - padre - pajaro - palacio - pared - parque - paseo - patio - perro - pescado - pez - piedra - pintor - piso - playa - plaza - policia - portal - portera - primo - provincia - pueblo - puente - puerta - puerto - raton - recibidor - region - restaurante - rico - rincon - rio - ruido - sala - senor - sereno - sierra - silla - sitio - sobrina - sobrino - sol - sotano - taller - tasca - teatro - terraza - tio/tia - tocadiscos - torero - toro - tren - vaca - vecino - ventana - viejo - zona -
+ les noms propres de lieu.

10. Lieux de llegar.

agua - almacen - aparador - arbol - armario - autobus - avenida - averia - balcon - barrio - brasero - brazo - cama - camino - camisa - campo - carcel - casa - ciego - cine - cita - ciudad - cocina - comedor - costa - cuarto - curso - dentista - despertador - desvan - doctor - dueno - edificio - ejercito - empleado - esposa - esposo - esquina - estacion - estatua - exito - flor - fuente - garage - habitacion - hospital - huesped - huevo - iglesia - jefe - labor - manzana - maquina - mar - mercado - mesa - meson - molino - montana - monte - mujer - orquesta - palacio - pared - parque - paseo - patio - piedra - piso - playa - plaza - portal - provincia - pueblo - puente - puerta - puerto - region - restaurante - rincon - rio - ruido - sala - sierra - sitio - sol - sotano - taller - tasca - teatro - terraza - tocadiscos - tren - vaca - ventana - zona -
+ les noms propres de lieu.

11. Lieux de subir.

agua - almacen - aparador - arbol - autobus - avenida - balcon - barrio - brazo - cama - camino - camisa - cartel - casa - ciudad - cocina - cuarto - curso - desvan - doctor - dueno - edificio - ejercito - empleado - esposa - esposo - estacion - estatua - exito - fuente - habitacion - huevo - iglesia - maquina - mesa - meson - molino - montana - monte - mujer - palacio - pared - parque - piso - playa - plaza - pueblo - puente - ruido - sala - sierra - silla - sitio - sol - taller - tasca - teatro - terraza - tren - ventana - zona -
+ les noms propres de lieu.

12. Lieux de nadar.

agua - amigo - arbol - avenida - brazo - ciudad - costa - empleado - esposa - esposo - esquina - exito - fabrica - fuente - gente - gitano - habitacion - huevo - mar - mujer - palacio - pared - playa - pueblo - puente - puerto - recipiente - rincon - rio - ruido - sitio - sol - terraza -
+ les noms propres de lieu.

13. Lieux de venir.

agua - almacen - amigo - aparador - arbol - armario - artista - asesino - autobus - avenida - averia - balcon - barrio - brasero - brazo - calle - campo - carcel - casa - ciego - cine - cita - ciudad -cocina - comedor - costa - cuarto - curso - dentista - despertador - desvan - doctor - dueno - edificio - ejercito - empleado - esposa - esposo - esquina - estacion - estatua - exito - fabrica - familia - flor - fuente - garage - gato - gente - gitano - habitacion - hospital - huesped - huevo - iglesia - jefe - labor - maquina - mar - medico - mercado - mesa - meson - molino - montana - monte - mudo - mujer - novio - orquesta - pajaro - palacio - pared - parque - paseo - patio - piedra - piso - playa - plaza - portal - provincia - pueblo - puente - puerta - puerto - recebidor - region - restaurante - rio - ruido - sala - sierra - silla - sitio - sol - sotano - taller - tasca - teatro - terraza - tocadiscos - torero - toro - tren - vaca - ventana - zona -
+ les noms propres de lieu.

14. Lieux de estar.

agua - aire - almacen - aparador - arbol - armario - autobus - avenida - averia - barrio - cama - camino - camisa - campo - carbon - carcel - casa - cine - cita - ciudad - cocina - comedor - comoda - costa - cuarto - curso - dentista - despertador - desvan - doctor - dueno - edificio - ejercito - esposa - esposo - esquina - estacion - exito - familia - fruta - fuente - garage - habitacion - hospital - iglesia - labor - lluvia - maceta - mar - mecanico - mefcado - mesa - meson - molino - montana - monte - mudo - mujer - ojo - oreja - orquesta - paisaje - pajaro - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerto - recibidor - recipiente - region - restaurante - rincon - rio - ropa - ruido - sala - semana - senor - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - vino - zapato - zona -
+ noms propres de lieu.

15. Lieux de situar(se).

agua - aire - almacen - aparador - arbol - armario - autobus - avenida - averia - barrio - camisa - campo - carcel - casa - cine - cita - cocina - comedor - comoda - costa - cuarto - dentista - despertador - desvan - doctor - dueno - edificio - esposa - esposo - esquina - estacion - exito - fabrica - familia - fruta - fuente - garage - habitacion - hospital - iglesia - labor - mar - mecanico - mercado - mesa - meson - molino - montana - monte - mujer - ojo - oreja - orquesta - paisaje - pajaro - palacio - parque - patio - piso - playa - plaza - portal - provincia - pueblo - puerto - recibidor - recipiente - region - restaurante - rincon - rio - ropa - ruido - sala - semana - senor - sierra - silla - sitio - sotano - taller - tasca - teatro - terraza - tren - vino - zapato - zona - + les noms propres de lieu.

A N N E X E 13.

SPECIFICATIONS DU MODULE DE GENERATION DE PHRASES.

Programme GENERATIONPHRASES.

Fonction : génère une phrase en espagnol selon le schéma
<déterminant> <sujet> <verbe à l'infinitif>
<préposition> <déterminant> <lieu>.
Entre chaque phrase générée, demande à l'utilisateur
s'il faut arrêter ou continuer.

Procédure CHARBYTE.

Arguments : - LISTMOT : tableau des caractères des mots.
- SOURCE : entier, indice dans "LISTMOT" de la première lettre du mot à charger.

Résultat : - DESTINATION : string où l'on charge le mot commençant à "LISTMOT [SOURCE] ".

Fonction : charge dans "DESTINATION" le mot dont l'indice dans "LISTMOT" est "SOURCE".

Procédure CHARGCLASSUP.

Argument : - SUPPLECLASSES : ("#5 : classesup.data") : fichier des classes supplémentaires.

Résultats : - SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- DERNCLASUP : dimension de "SUPPLEMENTAIRECLASSE".
- PASCREE : booléen, vrai si "SUPPLEMENTAIRECLASSE" n'a pas été chargé, faux sinon.

Fonction : charge le tableau "SUPPLEMENTAIRECLASSE" et met "PASCREE" à faux si "#5 : classesup.data" existe, sinon met "PASCREE" à vrai.

Procédure CHARGEMENTDICTIONNAIRE.

Arguments : - DICTIONNAIRE : ("#5 : dictio.data") : fichier dictionnaire.
- SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- DERNCLASUP : dimension de "SUPPLEMENTAIRECLASSE".

Résultats : -LISTMOT : tableau des caractères des mots du dictionnaire.
-LISTSUJET : tableau des caractéristiques des sujets.
-LISTLIEU : tableau des caractéristiques des lieux.
-LISTVERBE : tableau des caractéristiques des verbes.
-INDICE : tableau des indices de la première lettre de chaque mot du dictionnaire dans "LISTMOT".
-NBREMOT : dimension de "INDICE".
-NBRECAR : dimension de "LISTMOT".
-NBRESUJET : dimension de "LISTSUJET".
-NBREVERBE : dimension de "LISTVERBE".
-NBRELIEU : dimension de "LISTLIEU".
-PASCREE : booléen, vrai si "LISTMOT" et "INDICE" ne sont pas chargés, faux sinon.

Fonction : charge le fichier dictionnaire des mots dans les différents tableaux "LISTMOT", "INDICE", "LISTSUJET", "LISTVERBE" et "LISTLIEU", et met "PASCREE" à faux si "#5 : dictio.data " existe, sinon met "PASCREE" à vrai.

Procédure CONTARRET.

Arguments : - LIGNE : numéro de la ligne à partir de laquelle on efface.
- NBRE : numéro de la dernière ligne que l'on efface.

Résultat : booléen, vrai si on veut arrêter, faux si on veut continuer.

Fonction : demande à l'utilisateur en affichant cette demande à la ligne 23 s'il veut arrêter ou continuer.
Dans le premier cas, met "FINI" à vrai.
Dans le second cas, met "FINI" à faux et efface les lignes situées entre la ligne "LIGNE" et la ligne "NBRE" incluses.

Procédure CORRECTIONPHRASES.

Arguments : - PHRASECORRECTE : ("#5 : phracor.data") : fichier des modèles de classes sujets.
- NBCLASUJET : nombre de classes sujets existantes.
- NBREVERBE : nombre de verbes existants.
- NBCLALIEU : nombre de classes lieux existantes.

Résultats : - CORRECTSEM : matrice booléenne en trois dimensions des modèles de phrases correctes.
- PASCREE : booléen, vrai si "CORRECTSEM" n'a pas été chargé, faux sinon.

Fonction : charge la matrice booléenne en trois dimensions "CORRECTSEM" et met "PASCREE" à faux si "#5 : phracor.data" existe, sinon met "PASCREE" à vrai.

Procédure GENEDETERMINANT.

Arguments : -MASCULIN : booléen, vrai si "DETERMINANT" doit être accordé au masculin, faux sinon.
-PLURIEL : booléen, vrai si "DETERMINANT" doit être accordé au pluriel, faux sinon.

Résultat : - DETERMINANT : article défini ou indéfini ou possessif ou démonstratif.

Fonction : génère aléatoirement un déterminant espagnol, article défini, article indéfini, adjectif possessif ou adjectif démonstratif, accordés en genre et en nombre selon les valeurs de "MASCULIN" et "PLURIEL" et le place dans "DETERMINANT".

Procédure GENELIEU.

Arguments : - NBRELIEU : dimension de "LISTLIEU".
- LISTLIEU : tableau des caractéristiques des lieux.
- LISTMOT : tableau des caractères des mots du dictionnaire.
- CLASSESUJET : numéro de la classe du sujet de la phrase à générer.
- INDV : numéro du verbe de la phrase à générer.
- CORRECTSEM : matrice booléenne des modèles de phrases correctes.

Résultats : - LIEU : lieu généré.
- DETLIEU : déterminant de "LIEU".

Fonction : génère aléatoirement un lieu, convenant pour la classe sujet "CLASSESUJET" et le verbe "INDV", et son déterminant.

Procédure GENESUJET.

Arguments : - NBRESUJET : dimension de "LISTSUJET".
- LISTSUJET : tableau des caractéristiques des sujets.
- LISTMOT : tableau des caractères des mots du dictionnaire.

Résultats : - SUJET : sujet généré.
- PLURIELSUJET : booléen, vrai si "SUJET" est au pluriel, faux sinon.
- DETSUJET : déterminant de "SUJET".
- CLASSESUJET : numéro de la classe de "SUJET".

Fonction : génère aléatoirement un sujet et son déterminant.

Procédure GENEVERBE.

Arguments : - NBREVERBE : dimension de "LISTVERBE".
- LISTVERBE : tableau des caractéristiques des verbes.
- LISTMOT : tableau des caractères des mots du dictionnaire.
- CLASSESUJET : numéro de la classe du sujet de la phrase à générer.
- CORRECTSEM : matrice booléenne des modèles de phrases correctes.

Résultats : - PREP : préposition de lieu.
- VERBE : verbe généré et convenant pour la classe sujet "CLASSESUJET".
- INDV : numéro de "VERBE".

Fonction : génère aléatoirement un verbe, convenant pour la classe sujet "CLASSESUJET", et la préposition adéquate.

Procédure LOADCLASSEMOT.

Argument : - CLASSEMOT : ("#5 : listclas.data") : fichier des classes de mots existantes.

Résultats : - NBRECLASSE : nombre de classes de mots existantes.
- NBCLASUJET : nombre de classes sujets existantes.
- NBCLALIEU : nombre de classes lieux existantes.
- PASCREE : booléen, vrai si "CLASSEMOT" existe sur disquette, faux sinon.

Fonction : donne leur valeur à "NBRECLASSE", "NBCLASUJET", "NBCLALIEU" et "PASCREE".

Procédure MISEPLUR.

Arguments : - PLURIEL : vrai s'il faut mettre "ENTREE" au pluriel, faux sinon.
- ENTREE : string, mot espagnol à mettre au pluriel.

Résultat : - ENTREE : string, mot espagnol mis au pluriel.

Fonction : met "ENTREE", mot espagnol, au pluriel si "PLURIEL" est vrai selon les règles de grammaire espagnole : ajoute "s" si la dernière lettre de "ENTREE" au singulier est une voyelle, ajoute "es" si la dernière lettre de "ENTREE" est une consonne.
Si "ENTREE" est terminé par "z" au singulier, remplace ce"z" par "ces" et si "ENTREE" compte plus de 4 lettres et se termine par "s", ne le modifie pas.

```
(*  ANNEXE 14:LISTING DU MODULE DE GENERATION DE PHRASES.
  ================================================================  *)

(*$S+*)
PROGRAM GENERATIONPHRASES;
(*=========================*)
USES APPLESTUFF;

CONST VIDE='                                                  ';
      DIMLIST=60;
      DIMLIVERBE=16;
      MOITIENBREMOT=50;
      BORNMAT=16;
      DIMCAR=1200;
TYPE CHAINE=STRING[20];
     STR1=STRING[1];
     SEMAN=PACKED RECORD
       VALEUR:BOOLEAN;
       SUJET:INTEGER;
       VERBE:INTEGER;
       LIEU:INTEGER
      END;
     CORRECT=PACKED ARRAY[1..BORNMAT,1..BORNMAT,0..BORNMAT]OF BOOLEAN;
     TABCAR=PACKED ARRAY[1..DIMCAR] OF CHAR;
     TYPMOT=PACKED RECORD
       GENRE:BOOLEAN;
       POSPLUR:BOOLEAN;
       POSDET:BOOLEAN;
       MOT:CHAINE;
       CLASSE:INTEGER;
       PTRCLASSE:INTEGER
      END;
     CLASSE=RECORD
       NUMERO:INTEGER;
       TITRE:CHAINE;
      END;
     CLASSUP=RECORD
       NUMERO:INTEGER;
       CLASPTR:INTEGER
      END;
     LICLASSUP=PACKED ARRAY[0..MOITIENBREMOT] OF CLASSUP;
     CARACVERBE=RECORD
       INDICE:INTEGER;
       CLASSE:INTEGER
      END;
     CARACMOT=PACKED RECORD
       GENRE:BOOLEAN;
       POSPLUR:BOOLEAN;
       POSDET:BOOLEAN;
       INDICE:INTEGER;
       CLASSE:INTEGER
      END;
     LISTE=PACKED ARRAY[1..DIMLIST] OF CARACMOT;

VAR SUJET,DETSUJET,VERBE,LIEU,DETLIEU,PREPOSITION:CHAINE;
    SUPPLEMENTAIRECLASSE:LICLASSUP;
    LISTMOT:TABCAR;
    LISTSUJET,LISTVERBE,LISTLIEU:LISTE;
    INDV,CLASSESUJET,I,DERNCLASSUP,NBRECLASSE,NBCLASUJET,NBCLALIEU,NBREVERBE,
     NBRESUJET,NBRELIEU,NBRECAR,LIGNE:INTEGER;
    PASCREE,PLURIELSUJET,FINI:BOOLEAN;
    CORRECTSEM:CORRECT;

PROCEDURE LOADCLASSEMOT(VAR NBRECLASSE,NBCLASUJET,NBCLALIEU:INTEGER;
                            VAR PASCREE:BOOLEAN);
VAR IND:INTEGER;
    CLASSEMOT:FILE OF CLASSE;
    CLASEXIST:CLASSE;
```

```
BEGIN
  IND:=0;
  NBCLALIEU:=0;
  NBCLASUJET:=0;
  (*$I-*)
  RESET(CLASSEMOT,'#5:LISTCLAS.DATA');
  IF IORESULT = 0
   THEN
    BEGIN
     (*$I+*)
     PASCREE:=FALSE;
     WHILE NOT EOF(CLASSEMOT)DO
       BEGIN
         IND:=IND+1;
         CLASEXIST:=CLASSEMOT^;
         IF CLASEXIST.NUMERO<=100
           THEN NBCLASUJET:=NBCLASUJET+1
           ELSE NBCLALIEU:=NBCLALIEU+1;
         GET(CLASSEMOT)
       END
    END
   ELSE PASCREE:=TRUE;
 CLOSE(CLASSEMOT,LOCK);
 NBRECLASSE:=IND
END;

PROCEDURE CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE:LICLASSUP;
        DERNCLASSUP:INTEGER;VAR LISTMOT:TABCAR;VAR LISTSUJET,LISTLIEU,LISTVERBE:
        LISTE;VAR NBRECAR,NBRESUJET,NBREVERBE,
        NBRELIEU:INTEGER;VAR PASCREE:BOOLEAN);
VAR IND,INDS,INDV,INDL,CLASSESUIVANTE:INTEGER;
    ENTREE:TYPMOT;
    DICTIONNAIRE:FILE OF TYPMOT;
BEGIN
  INDS:=0;INDV:=0;INDL:=0;
  IND:=1;
  LISTMOT[IND]:='@';
  (*$I-*)
  RESET(DICTIONNAIRE,'#5:DICTIO.DATA');
  IF IORESULT=0
   THEN
    BEGIN
     (*$I+*)
     PASCREE:=FALSE;
     WHILE NOT EOF(DICTIONNAIRE) DO
       BEGIN
         IND:=IND+1;
         ENTREE:=DICTIONNAIRE^;
         IF ENTREE.CLASSE<=100
          THEN
           BEGIN
             INDS:=INDS+1;
             LISTSUJET[INDS].INDICE:=IND;
             LISTSUJET[INDS].GENRE:=ENTREE.GENRE;
             LISTSUJET[INDS].POSPLUR:=ENTREE.POSPLUR;
             LISTSUJET[INDS].POSDET:=ENTREE.POSDET;
             LISTSUJET[INDS].CLASSE:=ENTREE.CLASSE
           END
          ELSE IF ENTREE.CLASSE<=200
           THEN
            BEGIN
              INDL:=INDL+1;
              LISTLIEU[INDL].INDICE:=IND;
              LISTLIEU[INDL].GENRE:=ENTREE.GENRE;
              LISTLIEU[INDL].POSPLUR:=ENTREE.POSPLUR;
              LISTLIEU[INDL].POSDET:=ENTREE.POSDET;
              LISTLIEU[INDL].CLASSE:=ENTREE.CLASSE
            END
```

```
            ELSE
             BEGIN
               INDV:=INDV+1;
               LISTVERBE[INDV].INDICE:=IND;
               LISTVERBE[INDV].CLASSE:=ENTREE.CLASSE
             END;
          CLASSESUIVANTE:=ENTREE.PTRCLASSE;
          WHILE CLASSESUIVANTE>0 DO
            BEGIN
              IF SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO<=100
               THEN
                BEGIN
                  INDS:=INDS+1;
                  LISTSUJET[INDS].INDICE:=IND;
                  LISTSUJET[INDS].GENRE:=ENTREE.GENRE;
                  LISTSUJET[INDS].POSPLUR:=ENTREE.POSPLUR;
                  LISTSUJET[INDS].POSDET:=ENTREE.POSDET;
                  LISTSUJET[INDS].CLASSE:=
                      SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO
                END
               ELSE
                BEGIN
                  INDL:=INDL+1;
                  LISTLIEU[INDL].INDICE:=IND;
                  LISTLIEU[INDL].GENRE:=ENTREE.GENRE;
                  LISTLIEU[INDL].POSPLUR:=ENTREE.POSPLUR;
                  LISTLIEU[INDL].POSDET:=ENTREE.POSDET;
                  LISTLIEU[INDL].CLASSE:=
                    SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO
                END;
              CLASSESUIVANTE:=SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].CLASPTR
             END;
            MOVELEFT(ENTREE.MOT[1],LISTMOT[IND],LENGTH(ENTREE.MOT));
            IND:=IND+LENGTH(ENTREE.MOT);
            LISTMOT[IND]:='@';
            GET(DICTIONNAIRE)
        END;(*WHILE NOT EOF*)
      CLOSE(DICTIONNAIRE,LOCK)
     END(*IORESULT*)
    ELSE PASCREE:=TRUE;
   LISTMOT[IND+1]:='Z';
   LISTMOT[IND+2]:='Z';
   LISTMOT[IND+3]:='@';
   NBRECAR:=IND+3;
   NBRESUJET:=INDS;
   NBREVERBE:=INDV;
   NBRELIEU:=INDL
END;(*CHARGEMENTDICTIONNAIRE*)

PROCEDURE CHARGCLASSUP(VAR SUPPLEMENTAIRECLASSE:LICLASSUP;VAR DERNCLASSUP:
                          INTEGER;VAR PASCREE:BOOLEAN);
VAR IND:INTEGER;
    SUPPLECLASSES:FILE OF CLASSUP;
BEGIN
  IND:=0;
  (*$I-*)
  RESET(SUPPLECLASSES,'#5:CLASSESUP.DATA');
  IF IORESULT = 0
   THEN
    BEGIN
     (*$I+*)
     PASCREE:=FALSE;
     WHILE NOT EOF(SUPPLECLASSES)DO
      BEGIN
       IND:=IND+1;
       SUPPLEMENTAIRECLASSE[IND]:=SUPPLECLASSES^;
       GET(SUPPLECLASSES)
      END
```

```
    END
   ELSE PASCREE:=TRUE;
  CLOSE(SUPPLECLASSES,LOCK);
  DERNCLASSUP:=IND
END;(*CHARGCLASSUP*)

PROCEDURE CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU:INTEGER;VAR
                                    CORRECTSEM:CORRECT;VAR PASCREE:BOOLEAN);
VAR INDL,INDV,INDS:INTEGER;
    PHRASECORRECTE:FILE OF SEMAN;
BEGIN
  FOR INDS:=1 TO NBCLASUJET DO
   FOR INDV:=1 TO NBREVERBE DO
    FOR INDL:=0 TO NBCLALIEU DO
       CORRECTSEM[INDS,INDV,INDL]:=FALSE;
  (*$I-*)
  RESET(PHRASECORRECTE,'#5:PHRACOR.DATA');
  IF IORESULT = 0   (*$I+*)
   THEN
    BEGIN
     PASCREE:=FALSE;
     WHILE NOT EOF(PHRASECORRECTE)DO
       BEGIN
        WITH PHRASECORRECTE^ DO CORRECTSEM[SUJET,VERBE,LIEU]:=VALEUR;
        GET(PHRASECORRECTE)
       END
    END
  ELSE PASCREE:=TRUE;
 CLOSE(PHRASECORRECTE,LOCK)
END;(*CORRECTIONPHRASES*)

PROCEDURE CHARBYTE(LISTMOT:TABCAR;SOURCE:INTEGER;VAR DESTINATION:CHAINE);
VAR LONGUEUR:INTEGER;
BEGIN
  MOVELEFT(LISTMOT[SOURCE-1],DESTINATION,20);
  LONGUEUR:=20;
  MOVELEFT(LONGUEUR,DESTINATION,1);
  LONGUEUR:=POS('@',DESTINATION)-1;
  MOVELEFT(LONGUEUR,DESTINATION,1);
END;

PROCEDURE CONTARRET(LIGNE,NBRE:INTEGER;VAR FINI:BOOLEAN);
VAR REP:CHAR;
    I:INTEGER;
BEGIN
 READ(KEYBOARD,REP);
 WHILE (REP<>'A')AND(REP<>'C')DO
  BEGIN
   GOTOXY(40,23);
   READ(KEYBOARD,REP)
  END;
 IF REP='A'
  THEN FINI:=TRUE
  ELSE
   BEGIN
    GOTOXY(0,LIGNE);
    FINI:=FALSE;
    FOR I:=LIGNE TO NBRE DO
     BEGIN
      GOTOXY(0,I);
      WRITE(VIDE)
     END
   END;
END;(*CONTARRET*)

PROCEDURE MISEPLUR(VAR ENTREE:CHAINE);
VAR VOYELLES:SET OF 'A'..'U';
BEGIN
```

```
    VOYELLES:=['A','E','I','O','U'];
    IF ENTREE[LENGTH(ENTREE)] IN VOYELLES
     THEN ENTREE:=CONCAT(ENTREE,'S')
     ELSE
      BEGIN
       IF ENTREE[LENGTH(ENTREE)]='Z'
        THEN
         BEGIN
          DELETE(ENTREE,LENGTH(ENTREE),1);
          ENTREE:=CONCAT(ENTREE,'C')
         END;
       IF NOT(ENTREE[LENGTH(ENTREE)]='S')OR NOT(LENGTH(ENTREE)>4)
           THEN ENTREE:=CONCAT(ENTREE,'ES')
      END
END;(*MISEPLUR*)

PROCEDURE GENEDETERMINANT(MASCULIN,PLURIEL:BOOLEAN;VAR DETERMINANT:CHAINE);
VAR INDART,INDEM,INDPOS:INTEGER;
BEGIN
 RANDOMIZE;
 INDART:=1 + RANDOM MOD(4);
 CASE INDART OF
  1:IF MASCULIN
       THEN IF NOT PLURIEL THEN DETERMINANT:='EL'
                           ELSE DETERMINANT:='LOS'
       ELSE IF NOT PLURIEL THEN DETERMINANT:='LA'
                           ELSE DETERMINANT:='LAS';
  2:IF NOT PLURIEL THEN IF MASCULIN THEN DETERMINANT:='UN'
                                    ELSE DETERMINANT:='UNA'
                    ELSE DETERMINANT:='';
  3:BEGIN
     RANDOMIZE;
     INDEM:=1+RANDOM MOD (3);
     IF MASCULIN
      THEN IF NOT PLURIEL
           THEN CASE INDEM OF
                 1:DETERMINANT:='ESTE';
                 2:DETERMINANT:='ESE';
                 3:DETERMINANT:='AQUEL'
                 END(*CASE*)
           ELSE CASE INDEM OF
                 1:DETERMINANT:='ESTOS';
                 2:DETERMINANT:='ESOS';
                 3:DETERMINANT:='AQUELLOS'
                 END(*CASE*)
      ELSE IF NOT PLURIEL
           THEN CASE INDEM OF
                 1:DETERMINANT:='ESTA';
                 2:DETERMINANT:='ESE';
                 3:DETERMINANT:='AQUELLA'
                 END(*CASE*)
           ELSE CASE INDEM OF
                 1:DETERMINANT:='ESTAS';
                 2:DETERMINANT:='ESAS';
                 3:DETERMINANT:='AQUELLAS'
                 END(*CASE*)
    END;
  4:BEGIN
     RANDOMIZE;
     INDPOS:=1+RANDOM MOD (5);
     CASE INDPOS OF
      1:IF NOT PLURIEL THEN DETERMINANT:='MI'
                       ELSE DETERMINANT:='MIS';
      2:IF NOT PLURIEL THEN DETERMINANT:='TU'
                       ELSE DETERMINANT:='TUS';
      3:IF NOT PLURIEL THEN DETERMINANT:='SU'
                       ELSE DETERMINANT:='SUS';
      4:IF MASCULIN
```

```
          THEN IF NOT PLURIEL THEN DETERMINANT:='NUESTRO'
                              ELSE DETERMINANT:='NUESTROS'
          ELSE IF NOT PLURIEL THEN DETERMINANT:='NUESTRA'
                              ELSE DETERMINANT:='NUESTRAS';
       5:IF MASCULIN
          THEN IF NOT PLURIEL THEN DETERMINANT:='VUESTRO'
                              ELSE DETERMINANT:='VUESTROS'
          ELSE IF NOT PLURIEL THEN DETERMINANT:='VUESTRA'
                              ELSE DETERMINANT:='VUESTRAS';
       END(*CASE*)
      END(*4*)
     END;(*CASE*)
END;(*GENERATIONDETERMINANT*)

PROCEDURE GENESUJET(NBRESUJET:INTEGER;LISTSUJET:LISTE;LISTMOT:TABCAR;VAR SUJET:
                    CHAINE;VAR PLURIELSUJET:BOOLEAN;
                    VAR DETSUJET:CHAINE;VAR CLASSESUJET:INTEGER);
VAR I,INDCAR,INDS,NBRE:INTEGER;
BEGIN
  RANDOMIZE;
  INDS:=1+RANDOM MOD (NBRESUJET);
  INDCAR:=LISTSUJET[INDS].INDICE;
  I:=1;
  CHARBYTE(LISTMOT,INDCAR,SUJET);
  PLURIELSUJET:=FALSE;
  IF LISTSUJET[INDS].POSPLUR
    THEN
      BEGIN
        RANDOMIZE;
        NBRE:=1+RANDOM MOD (2);
        IF NBRE = 1 THEN
                    BEGIN
                     PLURIELSUJET:=TRUE;
                     MISEPLUR(SUJET)
                    END;
      END;
  IF LISTSUJET[INDS].POSDET
    THEN GENEDETERMINANT(LISTSUJET[INDS].GENRE,PLURIELSUJET,DETSUJET);
  CLASSESUJET:=LISTSUJET[INDS].CLASSE;
END;(*GENESUJET*)

PROCEDURE GENEVERBE(NBREVERBE:INTEGER;LISTVERBE:LISTE;LISTMOT:TABCAR;
                    CLASSESUJET:INTEGER;CORRECTSEM:CORRECT;VAR PREP:CHAINE;
                    VAR VERBE:CHAINE;VAR INDV:INTEGER);
VAR NBRE,INDCAR,I:INTEGER;
BEGIN
 REPEAT
   RANDOMIZE;
   INDV:=1 + RANDOM MOD(NBREVERBE)
  UNTIL CORRECTSEM[CLASSESUJET,INDV,0];
 CASE LISTVERBE[INDV].CLASSE OF
   201:PREP:='EN';
   301:PREP:='A';
   401:BEGIN
         RANDOMIZE;
         NBRE:=1 + RANDOM MOD(2);
         IF NBRE = 1 THEN PREP:='EN'
                     ELSE PREP:='A'
       END
  END;(*CASE*)
 INDCAR:=LISTVERBE[INDV].INDICE;
 I:=1;
 CHARBYTE(LISTMOT,INDCAR,VERBE);
END;(*GENEVERBE*)

PROCEDURE GENELIEU(NBRELIEU:INTEGER;LITSLIEU:LISTE;LITSMOT:TABCAR;CLASSESUJET,
                   INDV:INTEGER;CORRECTSEM:CORRECT;VAR LIEU,DETLIEU:CHAINE);
VAR INDL,INDCAR,I,NBRE:INTEGER;
```

```
    PLURIELIEU:BOOLEAN;
BEGIN
 REPEAT
   RANDOMIZE;
   INDL:=1 + RANDOM MOD(NBRELIEU)
  UNTIL CORRECTSEM[CLASSESUJET,INDV,LISTLIEU[INDL].CLASSE-100];
 INDCAR:=LISTLIEU[INDL].INDICE;
 I:=1;
 CHARBYTE(LISTMOT,INDCAR,LIEU);
 PLURIELIEU:=FALSE;
 IF LISTLIEU[INDL].POSPLUR
  THEN
   BEGIN
    RANDOMIZE;
    NBRE:=1 + RANDOM MOD(2);
    IF NBRE=1 THEN
     BEGIN
      PLURIELIEU:=TRUE;
      MISEPLUR(LIEU)
     END
   END;
 IF LISTLIEU[INDL].POSDET THEN GENEDETERMINANT(LISTLIEU[INDL].GENRE,PLURIELIEU,
            DETLIEU);
END;(*GENELIEU*)


(* PROGRAMME PRINCIPAL
========================  *)

BEGIN
  PAGE(OUTPUT);
  GOTOXY(0,0);
  WRITE('GENERATION ALEATOIRE DE PHRASES ESPAGNOLES');
  GOTOXY(0,1);FOR I:=1 TO 42 DO WRITE('-');
  CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASCREE);
  CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE,DERNCLASSUP,LISTMOT,LISTSUJET,
     LISTLIEU,LISTVERBE,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,PASCREE);
  LOADCLASSEMOT(NBRECLASSE,NBCLASUJET,NBCLALIEU,PASCREE);
  CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU,CORRECTSEM,PASCREE);
  LIGNE:=2;
  FINI:=FALSE;
  IF NOT PASCREE THEN
   WHILE NOT FINI DO
    BEGIN
     IF LIGNE>21
      THEN LIGNE:=2
      ELSE LIGNE:=LIGNE+1;
     GENESUJET(NBRESUJET,LISTSUJET,LISTMOT,SUJET,PLURIELSUJET,DETSUJET,
                CLASSESUJET);
     GENEVERBE(NBREVERBE,LISTVERBE,LISTMOT,CLASSESUJET,CORRECTSEM,PREPOSITION,
               VERBE,INDV);
     GENELIEU(NBRELIEU,LISTLIEU,LISTMOT,CLASSESUJET,INDV,CORRECTSEM,LIEU,
               DETLIEU);
     GOTOXY(0,LIGNE);FOR I:=1 TO 2 DO WRITELN(VIDE);
     GOTOXY(0,LIGNE);
     WRITE(DETSUJET,' ',SUJET,' ',VERBE,' ',PREPOSITION,' ',DETLIEU,' ',LIEU);
     LIGNE:=LIGNE+2;
     CONTARRET(LIGNE,LIGNE+3,FINI)
    END
   ELSE
    BEGIN
     GOTOXY(0,12);
     WRITE('PAS DE FICHIERS SUR LA DISQUETTE!')
    END
 END.
```

A N N E X E 15.

SPECIFICATIONS DU SYSTEME DE GESTION DE FICHIERS.

Programme GESMOT.

Fonction : permet à l'utilisateur de gérer les fichiers dictionnaire et de modèles de phrases correctes. Les actions permises sur le dictionnaire sont :

- l'établissement de listings à l'écran et à l'imprimante.
- l'ajout de mots.
- la modification de mots.
- la suppression de mots.
- la création de classes de mots.

Les actions permises sur le fichier de modèles de phrases correctes sont :

- l'établissement de listings à l'écran et à l'imprimante.
- l'ajout de modèles de phrases correctes.
- la suppression de modèles de phrases correctes.

Procédure ACCUEIL.

Fonction : explique à l'utilisateur l'utilité de "GESDICTIO" grâce à deux écrans.

Procédure ACCUEIL 1.

(interne à ACCUEIL).

Fonction : imprime le premier écran de "ACCUEIL".

Procédure ACCUEIL 2.

(interne à ACCUEIL).

Fonction : imprime le second écran de "ACCUEIL".

Procédure AJOUT.

Fonction : donne à l'utilisateur l'occasion d'ajouter un par un des mots et leurs caractéristiques dans le dictionnaire.

Procédure CORPAJOUT.

(interne à AJOUT).

Fonction : ajoute au dictionnaire un mot et ses caractéristiques, s'il ne s'y trouve pas déjà. Dans ce dernier cas, le signale à l'utilisateur. Termine toujours par la question "VOULEZ-VOUS ARRETER OU CONTINUER ?" affichée à la ligne 23.

Procédure AJOUPHRASES.

Fonction : donne à l'utilisateur l'occasion d'ajouter des modèles de phrases.

Procédure CORPAJOUPHRASES.

(interne à AJOUPHRASES).

Fonction : ajoute un modèle de phrases donné par l'utilisateur dans "CORRECTSEM".

Procédure AJOUSUJET.

(interne à CORPAJOUPHRASES).

Résultat : - INDS : numéro de classe sujet.

Fonction : reçoit dans "INDS" le numéro de la classe sujet du modèle de phrases.

Procédure AJOUVERBE.

(interne à CORPAJOUPHRASES).

Résultat : - INDV : numéro de verbe.

Fonction : reçoit dans "INDV" le numéro du verbe du modèle de phrases.

Procédure AJOULIEU.

(interne à CORPAJOUPHRASES).

Résultat : - INDL : numéro de classe lieu diminué de 100.

Fonction : reçoit dans "INDL" le numéro de la classe lieu du modèle de phrases diminué de 100.

Procédure AJOUTDICT.

Arguments : - ENTREE : mot à ajouter.
- INDAP : numéro d'ordre de "ENTREE" dans "LISTMOT" et "INDICE".

Arguments et Résultats : - LISTMOT : tableau des caractères des mots du dictionnaire.
- INDICE : tableau des indices de la première lettre de chaque mot de "LISTMOT".
- LISTSUJET : tableau des caractéristiques des sujets.
- LISTVERBE : tableau des caractéristiques des verbes.
- LISTLIEU : tableau des caractéristiques des lieux.
- SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- NBREMOT : dimension de "INDICE".
- NBRECAR : dimension de "LISTMOT.
- NBRESUJET : dimension de "LISTSUJET".
- NBREVERBE : dimension de "LISTVERBE".
- NBRELIEU : dimension de "LISTLIEU".
- DERNCLASSUP : dimension de "SUPPLEMENTAIRECLASSE".

Fonction : ajoute dans "LISTMOT" et "INDICE" le mot "ENTREE" en "INDAP^{ème}" position et, selon le cas, l'ajoute dans "LISTVERBE" ou "LISTSUJET" et/ou "LISTLIEU".

Procédure AJOUTLISTE.

(interne à AJOUTDICT).

Arguments : -ENTREE : mot à ajouter.
-MOTTAP : numéro d'ordre dans "ENS" de l'endroit où insérer "ENTREE".

-INDCARAP : indice du premier caractère de "ENTREE" dans "LISTMOT".
-NROCLASSE : numéro de la classe de "ENTREE".

Arguments et Résultats : - ENS : liste où on ajoute "ENTREE"
- NBRE : dimension de "ENS"

Fonction : ajoute en "MOTTAP^{ème}" position dans "ENS" le mot "ENTREE" avec "INDCARAP" comme partie "INDICE" et "NROCLASSE" comme partie "CLASSE".

Procédure AJOUTVERBE.

(interne à AJOUTDICT).

Arguments : - ENTREE : verbe que l'on ajoute.
- VERBAB : numéro d'ordre dans "LISTVERBE" de l'endroit où insérer "ENTREE".
- INDCARAP : indice du premier caractère de "ENTREE" dans "LISTMOT".

Fonction : ajoute en "VERBAB^{ème}" position dans "LISTVERBE" le mot "ENTREE" avec "INDCARAP" comme partie "INDICE".

Procédure CHARBYTE.

Arguments : - LISTMOT : tableau des caractères des mots du dictionnaire.
- SOURCE : indice dans "LISTMOT" de la première lettre du mot à charger.
- LONGUEUR : nombre de caractères (et de bytes) du mot à chargèr.

Résultat : DESTINATION : chaîne où est chargé le mot ayant "LONGUEUR" caractères et commençant à l'indice "SOURCE" dans "LISTMOT".

Fonction : charge dans "DESTINATION" le mot dont l'indice du premier caractère dans "LISTMOT" est "SOURCE" et ayant "LONGUEUR" caractères.

Procédure CHARGCLASSUP.

Argument : SUPPLECLASSES : [illegible] ("#5 : classesup. data") : fichier des classes supplémentaires.

Résultats : - SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- DERNCLASSUP : dimension de "SUPPLEMENTAIRECLASSE".
- PASCREE : booléen, vrai si "SUPPLEMENTAIRECLASSE" n'a pas été chargé, faux sinon.

Fonction : charge le tableau "SUPPLEMENTAIRECLASSE" et met "PASCREE" à faux si "#5 : classesup.data" existe, sinon met "PASCREE" à vrai.

Procédure CHARGEMENTDICTIONNAIRE.

Arguments : - DICTIONNAIRE ("#5 : dictio. data") : fichier dictionnaire.
- SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- DERNCLASUP : dimension de "SUPPLEMENTAIRECLASSE".

Résultats : - LISTMOT : tableau des caractères des mots du dictionnaire.
- LISTSUJET : tableau des caractéristiques des sujets.
- LISTLIEU : tableau des caractéristiques des lieux.
- LISTVERBE : tableau des caractéristiques des verbes.
- INDICE : tableau des indices de la première lettre de chaque mot du dictionnaire dans "LISTMOT".
- NBREMOT : dimension de "INDICE".
- NBRECAR : dimension de "LISTMOT".
- NBRESUJET : dimension de "LISTSUJET".
- NBREVERBE : dimension de "LISTVERBE".
- NBRELIEU : dimension de "LISTLIEU".
- PASCREE : booléen, vrai si "LISTMOT" et "INDICE" ne sont pas chargés, faux sinon.

Fonction : charge le fichier dictionnaire des mots dans les différents tableaux "LISTMOT", "INDICE", "LISTSUJET", "LISTVERBE" et "LISTLIEU", et met "PASCREE" à faux si "#5 : dictio. data" existe, sinon met "PASCREE" à vrai.

Procédure CHARGVERBE.

Résultats : - LISTMOT : tableau des caractères des mots du dictionnaire.
- INDICE : tableau des indices de la première lettre de chaque mot dans "LISTMOT".
- LISTVERBE : tableau des caractéristiques des verbes.
- NBREMOT : dimension de "INDICE".
- NBREVERBE : dimension de "LISTVERBE".
- PASCREE : booléen, vrai si "LISTMOT" et "INDICE" ne sont pas chargés; faux sinon.

Fonction : charge uniquement les verbes du dictionnaire dans les différents tableaux "LISTMOT", "INDICE" et "LISTVERBE", et met "PASCREE" à faux si"#5 : dictio.data" existe, sinon met "PASCREE" à vrai.

Procédure CHERCLASSE.

Arguments : - CLASEXIST : tableau des classes de mots existantes.
- NBRECLASSE : dimension de "CLASEXIST".
- INDL : numéro de la classe de mots que l'on recherche.

Résultat : SORTIE : classe (numéro et titre).

Fonction : recherche dans "CLASEXIST" la classe de numéro "INDL" et met son numéro et son titre dans "SORTIE".

Procédure CLASSEXIST.

Fonction : fait apparaître sur la totalité de l'écran la liste des classes de mots existantes précédée de ce titre : "LISTE DES CLASSES DE MOTS EXISTANTES".

Procédure CONSULTATION.

Fonction : donne à l'utilisateur l'occasion de consulter un par un les mots du dictionnaire et leurs caractéristiques.

Procédure CONVERSION.

Argument : - CARAC : chaîne de caractères .

Résultat : - ENTIER : nombre entier.

Fonction : si "CARAC" est la représentation en caractères d'un nombre entier, alors "ENTIER" est la conversion de "CARAC" en nombre entier;
sinon "ENTIER" est mis à zéro.

Procédure CONT.

Fonction : fait apparaître à la ligne 23 :"TAPEZ "C" POUR CONTINUER!" et attend que l'on ait tapé "C" avant d'effacer la ligne 23 et de continuer.

Procédure CONTARRET.

Arguments : - LIGNE : numéro de la ligne à partir de laquelle on efface.
- NBRE : numéro de la dernière ligne que l'on efface.

Résultat : booléen, vrai si on veut arrêter, faux si on veut continuer.

Fonction : demande à l'utilisateur en affichant cette demande à la ligne 23 s'il veut arrêter ou continuer.
Dans le premier cas, met "FINI" à vrai.
Dans le second cas, met "FINI" à faux et efface les lignes situées entre la ligne "LIGNE" et la ligne "NBRE" incluses.

Procédure CORRECTIONPHRASES.

Arguments : - PHRASECORRECTE ("#5 : phracor. data") : fichier des modèles de classes sujets.
- NBCLASUJET : nombre de classes sujets existantes.
- NBREVERBE : nombre de verbes existants.
- NBCLALIEU : nombre de classes lieux existantes.

Résultats : -CORRECTSEM : matrice booléenne en trois dimensions des modèles de phrases correctes.
-PASCREE : booléen, vrai si "CORRECTSEM" n'a pas été chargé, faux sinon.

Fonction : charge la matrice booléenne en trois dimensions "CORRECTSEM" et met "PASCREE" à faux si "#5 : phracor. data" existe, sinon met "PASCREE" à vrai.

Procédure CRECLASSE.

Fonction : donne à l'utilisateur l'occasion de créer des classes de mots en leur attribuant un numéro et un titre.

Procédure RECEPCHOIX.

(interne à CRECLASSE).

Fonction : donne l'occasion à l'utilisateur d'exprimer son choix en tapant "1", "2" ou "3" à la colonne 38 de la ligne 6 et valide ce choix.

Procédure ECRCLALI.

Arguments : - CLASEXIST : tableau des classes de mots existantes.
- NBRECLASSE : dimension de "CLASEXIST".
- COLONNE : numéro de la colonne où on commence à écrire.
- LIGNE : numéro de la ligne où on commence à écrire.

Fonction : écrit à partir de la colonne "COLONNE" de la ligne "LIGNE" la suite des numéros et des titres des classes lieux existantes jusqu'à la ligne 18 et continue si nécessaire à partir de la colonne 25 de la ligne 12.

Procédure ECRCLASSEX.

Arguments : - CLASEXIST : tableau des classes de mots existantes.
- NBRECLASSE : dimension de "CLASEXIST".
- LIMITE : limite vers le bas des lignes où on écrit.
- COLONNE : numéro de la colonne où on commence à écrire.
- LIGNE : numéro de la ligne où on commence à écrire.

Fonction : écrit "LISTE DES CLASSES DE MOTS EXISTANTES" à partir de la colonne "COLONNE" de la ligne "LIGNE", souligne ce titre puis écrit cette liste de numéros et de titres jusqu'à la ligne "LIMITE" et continue si nécessaire à partir de la colonne 20 de la ligne "LIGNE" puis si nécessaire de la colonne 42 de la ligne "LIGNE".

Procédure ALALIGNE.

(interne à ECRCLASSEX).

Fonction : si "LIGNE" > = "LIMITE"
alors "LIGNE" reprend sa valeur de début de "ECRCLASEX"
si "COLONNE" < 20
alors "COLONNE" prend la valeur 20
sinon "COLONNE" prend la valeur 42
sinon "LIGNE" est augmenté de 1

Procédure ECRCLASU.

Arguments : - CLASEXIST : tableau des classes de mots existantes.
- NBRECLASSE : dimension de "CLASEXIST".
- COLONNE : numéro de la colonne où on commence à écrire.
- LIGNE : numéro de la ligne où on commence à écrire.

Fonction : écrit à partir de la colonne "COLONNE" de la ligne "LIGNE" la suite des numéros et des titres des classes sujets existantes jusqu'à la ligne 18 et continue si nécessaire à partir de la colonne 25 de la ligne 12.

Procédure ECRIVERBE.

Arguments : - LISTVERBE : tableau des caractéristiques des verbes.
- LISTMOT : tableau des caractères des mots du dictionnaire.
- INDV : numéro d'ordre d'un verbe dans "LISTVERBE".

Fonction : écrit "verbe" puis le numéro "INDV" puis le"INDV"ème verbe de "LISTVERBE" caractère par caractère.

Procédure ENCORECLASSE.

Argument : LIGNE : ligne où la procédure affiche sa question.

Résultat : ENCORE : réponse de l'utilisateur à la question posée par la procédure.
- si = "O" : encore une classe
- si = "N" : plus de classe

Fonction : efface la ligne "LIGNE" et la ligne 23, affiche à la ligne "LIGNE" : "ENCORE UNE CLASSE (O/N) ?", lit la réponse et la valide.

Procédure GESMOTS.

Résultat : CHOIXMOT : expression du choix de l'utilisateur.

si = 1: consultation du dictionnaire
si = 2: ajout de mot(s)
si = 3: modification de mot(s)
si = 4: suppression de mot(s)
si = 5: consultation de mot(s)
si = 6: création de classe de mot(s)
si = 7: retour au menu général
si = 8: fin de "GESDICTIO".

Fonction : donne à l'utilisateur l'occasion de choisir entre

- consultation du dictionnaire
- ajout de mot(s)
- modofication de mot(s)
- suppression de mot(s)
- consultation de mot(s)
- création de classe de mot(s)
- retour au menu général
- fin de "GESDICTIO".

Procédure GESPHRASES.

Résultat : CHOIXPHRASES : expression du choix de l'utilisateur.
si = 1 : consultation de la liste des modèles de phrases correctes.
si = 2 : ajout de modèles de phrases correctes.
si = 3 : suppression de modèles de phrases correctes.
si = 4 : retour au menu général.
si = 5 : fin de "GESDICTIO".

Fonction : donne à l'utilisateur l'occasion de choisir entre
- consultation de la liste des modèles de phrases correctes.
- ajout de modèles de phrases correctes.
- suppression de modèles de phrases correctes.
- retour au menu général.
- fin de "GESDICTIO".

Procédure IMPRESCLAS.

Fonction : imprime la liste des classes de mots existantes.

Procédure IMPRESELEM.

(interne à IMPRESCLAS).

Arguments : - STRNRO : expression en caractères d'un nombre, en l'occurence un numéro de classe.
- INTITULE : chaîne de caractères exprimant ici le titre d'une classe.

Fonction : imprime respectivement les valeurs de "STRNRO" et "INTITULE" précédées de la valeur de titre de "TITRE" et séparées par trois blancs, et envoie un "carriage return".

Procédure LISTING.

Fonction : donne à choisir à l'utilisateur entre
- listing à l'écran du dictionnaire
- listing à l'écran de la liste des classes de mots existantes
- listing à l'imprimante du dictionnaire
- listing à l'imprimante de la liste des classes de mots existantes.

Procédure LISTECRAN.

(interne à LISTING).

Fonction : si "DICTIONNAIRE" a été chargé en mémoire centrale, affiche à l'écran les articles du dictionnaire en entier et séquentiellement.
Sinon, dit qu'il n'y a pas de mot dans la liste.

Procédure LISTIMPRIM.

(interne à LISTING).

Fonction : imprime les articles de "DICTIONNAIRE" l'un après l'autre, un par ligne et précédé de "MOT NRO" et du numéro d'ordre du mot si "DICTIONNAIRE" existe sur disquette; sinon, dit qu'il n'y a pas de mot dans la liste.

Procédure CORIMPRIM.

(interne à LISTIMPRIM).

Fonction : assure l'impression en séquences des articles de "DICTIONNAIRE".

Procédure LISTPHRASES.

Fonction : donne à choisir à l'utilisateur entre
- listing à l'écran des modèles de phrases existants.
- listing à l'écran de la liste des classes de mots existantes.
- listing à l'imprimante des modèles de phrases existants.
- listing à l'imprimante de la liste des classes de mots existantes.

Procédure REFUSIMPRES.

(interne à LISTPHRASES).

Arguments : - PASCLAS : booléen, vrai si "CLASEXIST" n'a pas été chargé; faux sinon.
- PASVERB : booléen, vrai si "LISTVERBE" n'a pas été chargé; faux sinon.
- PASPHRA : booléen, vrai si "CORRECTSEM" n'a pas été chargé; faux sinon.

Fonction : explique à l'utilisateur pourquoi il ne peut obtenir de listing.

Procédure LISTPHRAECR.

(interne à LISTPHRASES).

Fonction : si "CORRECTSEM", "CLASEXIST" et "LISTVERBE" ont été chargés, affiche à l'écran la liste des modèles de phrases existants ;
sinon, explique pourquoi cela est impossible.

Procédure LISTPHRAIM.

(interne à LISTPHRASES).

Fonction : si "CORRECTSEM", "CLASEXIST" et "LISTVERBE" ont été chargés, imprime la liste des modèles de phrases existants;
sinon, explique pourquoi cela est impossible.

Procédure LOADCLASSEMOT.

Argument : CLASSEMOT ("#5 : listclas. data") : fichier des classes de mots existantes.

Résultats :
- CLASEXIST : tableau des classes de mots existantes.
- NBRECLASSE : dimension de "CLASEXIST".
- NBCLASUJET : nombre de classes sujets.
- NBCLALIEU : nombre de classes lieux.
- LISTNROCLASSE : ensemble des numéros de classes de mots existantes.
- LINOCLASU : ensemble des numéros de classes sujets existantes.
- LINOCLALI : ensemble des numéros de classes lieux existantes.
- PASCREE : booléen, vrai si "CLASEXIST" n'a pas été chargé, faux sinon.

Fonction : charge le tableau "CLASEXIST" et met "PASCREE" à faux si "#5 : listclas. data" existe, sinon met "PASCREE" à vrai.

Procédure MENUGENERAL.

Résultat : CHOIX : expression du choix de l'utilisateur.
si = 1 : gestion du dictionnaire
si = 2 : gestion du modèle des phrases correctes
si = 3 : fin de "GESDICTIO".

Fonction : donne à l'utilisateur l'occasion de choisir entre
- gestion du dictionnaire
- gestion du modèle des phrases correctes
- fin de "GESDICTIO".

Procédure MODIFICATION.

Fonction : donne à l'utilisateur l'occasion de modifier un par un des mots du dictionnaire et leurs caractéristiques.

Procédure MODIVERBE.

(interne à MODIFICATION).

Arguments : - INDVERBE : numéro d'ordre d'un verbe dans "LISTVERBE".
- LIGNE : numéro de la ligne où l'on fait la modification.

Fonction : met la nouvelle version du "INDV"^{ème} verbe dans "NOUVEAU".

Procédure MODIMOT.

(interne à MODIFICATION).

Arguments : - ENS : tableau de caractéristiques de mots.
- DIM : dimension de "ENS".
- INDMOT : numéro d'ordre d'un mot dans "ENS".
- LIGNE : numéro de la ligne courante à partir de laquelle on modifie.

Fonction : met la nouvelle version du "INDMOT"^{ème} mot dans "NOUVEAU".

Procédure MODIPHONEME.

(interne à MODIMOT).

Fonction : garnit "NOUVEAU.MOT".

Procédure_MODIGENRE.

(interne à MODIMOT).

Fonction : garnit "NOUVEAU.GENRE".

Procédure_MODIPOSPLUR.

(interne à MODIMOT).

Fonction : garnit "NOUVEAU.POSPLUR".

Procédure_MODIPOSDET._

(interne à MODIMOT).

Fonction : garnit "NOUVEAU.POSDET".

Procédure_MODICLASSE._

(interne à MODIMOT).

Fonction : garnit "NOUVEAU.CLASSE".

Procédure_MODISUPPLECLASSE._

(interne à MODIMOT).

Fonction : garnit "NOUVEAU.PTRCLASSE" et éventuellement "SUPPLEMENTAIRECLASSE".

Procédure ORTHOVRAI.

(interne à MODIFICATION).

Fonction : assure la modification d'un mot du dictionnaire.

Procédure MOTCARAC.

Arguments : - ENS : tableau de caractéristiques de mots.
- DIM : dimension de "ENS".
- INDMOT : position d'un mot dans "ENS".
- LIGNE : numéro de la ligne après laquelle on écrit les caractéristiques du mot.

Fonction : écrit les caractéristiques du mot à partir de la ligne "LIGNE+1", c'est-à-dire son genre, sa possibilité de pluriel, sa possibilité de déterminant et son(ou ses) numéro(s) de classe.

Procédure RECEPNROCLASSE.

Arguments : - LISTNROCLASSE : ensemble des numéros de classes de mots existantes
- COLONNE : numéro de la colonne où l'utilisateur doit commencer à écrire.
- LIGNE : numéro de la ligne où l'utilisateur doit écrire.
- MODIF : booléen, vrai si la procédure est appelée dans le cadre de la procédure"MODIFICATION" et doit considérer le "return" seul comme un refus de modification, faux sinon.

Résultats : - NROCLASSE : numéro de classe.
- LONG : longueur en chiffres de "NROCLASSE".

Fonction : lit le numéro de classe à la colonne "COLONNE" de la ligne "LIGNE".
Si "MODIF" est vrai et que la longueur de ce numéro est nulle, - alors : met (-1) dans "NROCLASSE".
- sinon, met la version "integer" de ce numéro s'il existe dans "LISTNROCLASSE" dans "NROCLASSE" et met sa longueur en chiffres dans "LONG".

Procédure RECHERCHE.

Arguments : - LISTMOT : tableau des caractères des mots du dictionnaire.
- INDICE : tableau des indices de la première lettre de chaque mot du dictionnaire dans "LISTMOT".
- NBREMOT : dimension de "INDICE".
- ENTREE : mot que l'on recherche.

Résultats : - ORTHO : booléen, vrai si "ENTREE" est dans "LISTMOT", faux sinon.
- INDMOT : numéro d'ordre alphabétique dans "LISTMOT" et "INDICE" de "ENTREE" s'il s'y trouve, indéterminé sinon.
- BI : position dans "LISTMOT" et "INDICE" de "ENTREE" s'il s'y trouvait et qu'il n'y est pas (= endroit où l'y insérer), indéterminé sinon.

Fonction : recherche si "ENTREE" existe dans "LISTMOT".
- si oui : "ORTHO" est vrai et "INDMOT" est le numéro d'ordre de "ENTREE" dans "INDICE".
- sinon : "ORTHO" est faux et "BI" est le numéro d'ordre du mot qui lui est juste supérieur dans l'ordre alphabétique dans "LISTMOT".

Procédure RECHSUVERLIEU.

Arguments : - ENS : liste de caractéristiques de mots.
- NBRE : dimension de "ENS".
- INDCARAP : indice du mot recherché dans "LISTMOT".

Résultats : - ORTHO : booléen, vrai si "INDCARAP" est dans "ENS.INDICE" faux sinon.
- INDSUVERLIEU : indice de "INDCARAP" dans "ENS" s'il y est, indéterminé sinon.
- BI : numéro d'ordre dans "ENS" de "INDCARAP" s'il s'y trouvait et qu'il n'y est pas (= endroit où l'y insérer), indéterminé sinon.

Fonction : recherche si "INDCARAP" existe dans la partie "INDICE" de "ENS".
- si oui : "ORTHO" est vrai et "INDSUVERLIEU" est son numéro d'ordre dans "ENS".
- sinon : "ORTHO" est faux, et "BI" est le numéro d'ordre de l'indice qui lui est juste supérieur dans "ENS".

Procédure SAUVECLASSUP.

Arguments : - SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- DERNCLASSUP : dimension de "SUPPLEMENTAIRECLASSE".

Résultat : SUPPLECLASSES ("#5 : classesup. data") : fichier des classes supplémentaires.

Fonction : sauve "SUPPLEMENTAIRECLASSE" dans "SUPPLECLASSES" sur disquette.

Procédure SAUVEPHRASECORRECTE.

Arguments : - CORRECTSEM : matrice booléenne en trois dimensions des modèles de phrases correctes.
- NBCLASUJET : nombre de classes sujets existantes.
- NBREVERBE : nombre de verbes existants.
- NBCLALIEU : nombre de classes lieux existantes.

Résultat : PHRASECORRECTE ("#5 : phracor.data") : fichier des modèles de phrases correctes.

Fonction : sauve "CORRECTSEM" dans "phrasecorrecte" sur disquette.

Procédure SAUVETAGE.

Arguments : - LISTMOT : tableau des caractères des mots du dictionnaire.
- INDICE : tableau des indices du premier caractère de chaque mot dans "LISTMOT".
- LISTSUJET : tableau des caractéristiques des sujets.
- LISTVERBE : tableau des caractéristiques des verbes.
- LISTLIEU : tableau des caractéristiques des lieux.
- NBRECAR : dimension de "LISTMOT".
- NBREMOT : dimension de "INDICE".
- NBRESUJET : dimension de "LISTSUJET".
- NBREVERBE : dimension de "LISTVERBE".
- NBRELIEU : dimension de "LISTLIEU".

Résultats : - DICTIONNAIRE ("#5 : dictio.data") : fichier dictionnaire des mots et de leurs caractéristiques.
- SUPPLEMENTAIRECLASSE : tableau des classes supplémentaires.
- DERNCLASSUP : dimension de "SUPPLEMENTAIRECLASSE".

Fonction : sauve "LISTMOT", "LISTSUJET", "LISTVERBE" et "LISTLIEU" dans "DICTIONNAIRE" sur disquette.

Procédure SORTVERBE.

(interne à SAUVETAGE).

Arguments : - NBREVERBE : dimension de "LISTVERBE".
- INDV : numéro dans "LISTVERBE" du verbe à sauver.

Fonction : met le "INDV^{ème}" verbe dans "SORTIE" et ajoute 1 à "INDV".

Procédure SORTSUJET.

(interne à SAUVETAGE).

Arguments : - NBRESUJET : dimension de "LISTSUJET".
- INDS : numéro dans "LISTSUJET" du sujet à sauver.

Fonction : met le "INDS^{ème}" sujet dans "SORTIE" ainsi que les sujets suivants ,s'ils sont identiques, dans "SUPPLEMENTAIRECLASSE" et incrémente "INDS".

Procédure SORTSULIEU.

(interne à SAUVETAGE).

Arguments : - NBRESUJET : dimension de "LISTSUJET".
- NBRELIEU : dimension de "LISTLIEU".
- INDS : numéro dans "LISTSUJET" du sujet à sauver.
- INDL : numéro dans "LISTLIEU" du lieu à sauver.

Fonction : mettre le "INDS^{ème}" sujet et le "INDL^{ème}" lieu dans "SORTIE", mettre à jour "SUPPLEMENTAIRECLASSE" et incrémenter "INDS" et "INDL".

Procédure SORTLIEU.

(interne à SAUVETAGE).

Arguments : - NBRELIEU : dimension de "LISTLIEU".
- INDL : numéro dans "LISTLIEU" du lieu à sauver.

Fonction : met le "INDL^{ème}" lieu dans "SORTIE" ainsi que les lieux suivants, s'ils sont identiques, dans "SUPPLEMENTAIRECLASSE" et incrémente "INDL".

Procédure SAVECLASSEMOT.

Arguments : - CLASEXIST : tableau des classes de mots existantes.
- NBRECLASSE : dimension de "CLASEXIST".

Résultat : CLASSEMOT ("#5 : listclas.data") : fichier des classes de mots existantes.

Fonction : sauve "CLASEXIST" dans "CLASSEMOT" sur disquette.

Procédure SULIEUCARAC.

Arguments : - LISTLIEU : tableau de caractéristiques de lieux.
- NBRELIEU : dimension de "LISTLIEU".
- INDL : numéro d'ordre d'un lieu dans "LISTLIEU".

Fonction : écrit tous les numéros de classes de lieux se rapportant au même mot.

Procédure SUPPRESDICT.

Arguments : - ENTREE : mot que l'on supprime.
- INDMOT : position de "ENTREE" dans "INDICE" et "LISTMOT".

Arguments et Résultats : - LISTMOT : tableau des caractères des mots du dictionnaire.
- INDICE : tableau des indices de la première lettre de chaque mot du dictionnaire dans "LISTMOT".
- LISTSUJET : tableau des caractéristiques des sujets.
- LISTVERBE : tableau des caractéristiques des verbes.
- LISTLIEU : tableau des caractéristiques des lieux.
- NBREMOT : dimension de "INDICE".
- NBRECAR : dimension de "LISTMOT".
- NBRESUJET : dimension de "LISTSUJET".
- NBREVERBE : dimension de "LISTVERBE".
- NBRELIEU : dimension de "LISTLIEU".

Fonction : supprime le mot "ENTREE" qui est en "INDMOT^{ème}" position dans "LISTMOT" et "INDICE" et, selon le cas, le supprime de "LISTVERBE" ou "LISTSUJET" et/ou "LISTLIEU".

Procédure SUPPRELISTE.

(interne à SUPPRESDICT).

Arguments : - INDCAR : position dans "LISTMOT" du premier caractère de "ENTREE".
- MARK : position de "ENTREE" dans "ENS".

Arguments et Résultats : - ENS : tableau des caractéristiques de mots dans lequel se trouvent celles du mot à supprimer.

- DIMENS : dimension de "ENS".

Fonction : supprime dans "ENS" tous les éléments ayant comme partie "INDICE" la valeur "INDCAR".

Procédure SUPPRESPHRASES.

Fonction : donne à l'utilisateur l'occasion de supprimer des modèles de phrases.

Procédure CORPSUPPRESPHRASES.

(interne à SUPPRESPHRASES).

Fonction : supprime un modèle de phrases donné par l'utilisateur dans "CORRECTSEM".

Procédure SUPVERBE.

(interne à CORPSUPPRESPHRASES).

Résultat : - INDV : numéro de verbe.

Fonction : reçoit dans "INDV" le numéro du verbe du modèle de phrases.

Procédure SUPSUJET.

(interne à CORPSUPPRESPHRASES).

Résultat : - INDS : numéro de classe sujet.

Fonction : reçoit dans "INDS" le numéro de la classe sujet du modèle de phrases.

Procédure SUPLIEU.

(interne à CORPSUPPRESPHRASES).

Résultat : - INDL : numéro de classe lieu diminué de 100.

Fonction : reçoit dans "INDL" le numéro de la classe lieu du modèle de phrases diminué de 100.

Procédure SUPPRESSION.

Fonction : donne à l'utilisateur de supprimer un par un des mots du dictionnaire.

Procédure CONSUPPRES.

(interne à SUPPRESSION).

Arguments : - DEPOUI : nombre de lignes à ajouter à "LIGNE" si on supprime.
- DEPNON : nombre de lignes à ajouter à "LIGNE" si on ne supprime pas.

Fonction : demande à l'utilisateur "ETES-VOUS SUR QU'IL FAUT LE SUPPRIMER ?"
- si oui, supprime le mot et incrémente "LIGNE" de "DEPOUI".
- sinon, incrémente "LIGNE" de "DEPNON".

Procédure TERMINER.

Fonction : efface l'écran et fait apparaître au milieu :
"FIN DE LA GESTION DU DICTIONNAIRE" et "AU REVOIR".

Procédure VERBECARAC.

Arguments : - LISTVERBE : liste des caractéristiques des verbes.
- INDVERBE : position d'un verbe dans "LISTVERBE".
- LIGNE : numéro de la ligne après laquelle on écrit la caractéristique du verbe.

Fonction : écrit la caractéristique du verbe à la ligne "LIGNE+1", c'est-à-dire son numéro de classe et la préposition demandée.

Procédure VERBEX.

Arguments : - LISTVERBE : tableau des caractéristiques des verbes.
- LISTMOT : tableau des caractères des mots du dictionnaire
- NBREVERBE : dimension de "LISTVERBE".
- COLONNE : numéro de la colonne où on commence à écrire.
- LIGNE : numéro de la ligne où on commence à écrire.

Fonction : écrit à partir de la colonne "COLONNE" de la ligne "LIGNE" la liste des verbes existants.

```
(*   ANNEXE 16:LISTING DU SYSTEME DE GESTION DES FICHIERS.
  ================================================================  *)

(*$S+,N+*)
PROGRAM GESDICTIO;
(*================*)

USES (*$U *:U1.CODE*)U1,(*$U *:U2.CODE*)U2,(*$U *:U3.CODE*)U3,(*$U *:U4.CODE*)
       U4,(*$U *:U5.CODE*)U5,(*$U *:U6.CODE*)U6,(*$U APPLE2:U7.CODE*)U7,
       (*$U APPLE2:U7B.CODE*)U7B,(*$U APPLE2:U8.CODE*)U8,(*$U APPLE2:U8B.CODE*)
       U8B,(*$U APPLE2:U9.CODE*)U9,(*$U APPLE2:U10.CODE*)U10,
       (*$U APPLE2:U12.CODE*)U12,(*$U APPLE2:U13.CODE*)U13,
       (*$U APPLE2:U14.CODE*)U14;

VAR FIN:BOOLEAN;
    CHOIX,CHOIXMOT,CHOIXPHRASE:CHAR;

BEGIN
  (*$R U1,U2*)
  ACCUEIL;
  FIN:=FALSE;
  WHILE NOT FIN DO
    BEGIN
      MENUGENERAL(CHOIX);
      CASE CHOIX OF
        '1':BEGIN
              GESMOTS(CHOIXMOT);
              WHILE (CHOIXMOT<>'7')AND(CHOIXMOT<>'8')DO
                BEGIN
                  CASE CHOIXMOT OF
                    '1':LISTING;
                    '2':AJOUT;
                    '3':MODIFICATION;
                    '4':SUPPRESSION;
                    '5':CONSULTATION;
                    '6':CRECLASSE
                  END;(*CASE*)
                  GESMOTS(CHOIXMOT)
                END;
              IF CHOIXMOT='8'THEN FIN:=TRUE
            END;
        '2':BEGIN
              GESPHRASES(CHOIXPHRASES);
              WHILE(CHOIXPHRASES<>'4')AND(CHOIXPHRASES<>'5')DO
                BEGIN
                  CASE CHOIXPHRASES OF
                    '1':LISTPHRASES;
                    '2':AJOUPHRASES;
                    '3':SUPPRESPHRASES
                  END;(*CASE*)
                  GESPHRASES(CHOIXPHRASES)
                END;
              IF CHOIXPHRASES='5'THEN FIN:=TRUE
            END;
        '3':FIN:=TRUE
      END(*CASE*)
    END;(*WHILE NOT FINI*)
  TERMINER
END.
```

```
(*$S+*)

UNIT U1;

INTERFACE

CONST VIDE='                                                  ';
      DIMLIST=60;
      DIMLIVERBE=16;
      MOITIENBREMOT=50;
      DIMCLASEX=30;
      BORNMAT=16;
      DIMCAR=1200;
      DIMMOT=200;
TYPE CHAINE=STRING[20];
     STR1=STRING[1];
     SET150=SET OF 1..150;
     SET1=SET OF 1..100;
     SET51=SET OF 51..150;
     SET8=SET OF '1'..'8';
     SET6=SET OF '1'..'6';
     FONCTION=(S,V,D,L);
     SEMAN=PACKED RECORD
       VALEUR:BOOLEAN;
       SUJET:INTEGER;
       VERBE:INTEGER;
       LIEU:INTEGER
      END;
     CORRECT=PACKED ARRAY[1..BORNMAT,1..BORNMAT,0..BORNMAT]OF BOOLEAN;
     TABCAR=PACKED ARRAY[1..DIMCAR] OF CHAR;
     PTRMOT=PACKED RECORD
       PTRTYP:CHAR;
       INDLIS:INTEGER
      END;
     TABIND=PACKED ARRAY[1..DIMMOT] OF PTRMOT;
     TYPMOT=PACKED RECORD
       GENRE:BOOLEAN;
       POSPLUR:BOOLEAN;
       POSDET:BOOLEAN;
       MOT:CHAINE;
       CLASSE:INTEGER;
       PTRCLASSE:INTEGER
      END;
     CLASSE=RECORD
       NUMERO:INTEGER;
       TITRE:CHAINE;
      END;
     EXICLASSE=PACKED ARRAY[0..DIMCLASEX] OF CLASSE;
     CLASSUP=RECORD
       NUMERO:INTEGER;
       CLASPTR:INTEGER
      END;
     LICLASSUP=PACKED ARRAY[0..MOITIENBREMOT] OF CLASSUP;
     CARACVERBE=RECORD
       INDICE:INTEGER;
       CLASSE:INTEGER
      END;
     CARACMOT=PACKED RECORD
       GENRE:BOOLEAN;
       POSPLUR:BOOLEAN;
       POSDET:BOOLEAN;
       INDICE:INTEGER;
       CLASSE:INTEGER
      END;
     LISTE=PACKED ARRAY[1..DIMLIST] OF CARACMOT;

VAR SUPPLECLASSES:FILE OF CLASSUP;
    CLASSEMOT:FILE OF CLASSE;
```

```
    PHRASECORRECTE:FILE OF SEMAN;

PROCEDURE CHARGCLASSUP(VAR SUPPLEMENTAIRECLASSE:LICLASSUP;VAR DERNCLASSUP:
                       INTEGER;VAR PASCREE:BOOLEAN);
PROCEDURE LOADCLASSEMOT(VAR CLASEXIST:EXICLASSE;VAR NBRECLASSE,NBCLASUJET,
                        NBCLALIEU:INTEGER;VAR LISTNROCLASSE:SET150;VAR
                        LINOCLASU:SET1;VAR LINOCLALI:SET51;VAR PASCREE:BOOLEAN);

PROCEDURE CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU:INTEGER;VAR
                            CORRECTSEM:CORRECT;VAR PASCREE:BOOLEAN);
PROCEDURE CHARBYTE(LISTMOT:TABCAR;SOURCE:INTEGER;VAR LONGUEUR:INTEGER;VAR
                   DESTINATION:CHAINE);
PROCEDURE RECHERCHE(LISTMOT:TABCAR;INDICE:TABIND;NBREMOT:INTEGER;ENTREE:CHAINE;
                    VAR ORTHO:BOOLEAN;VAR INDMOT,BI:INTEGER);
PROCEDURE RECHSUVERLIEU(ENS:LISTE;NBRE,INDCARAP:INTEGER;VAR ORTHO:BOOLEAN;VAR
                        INDSUVERLIEU,BI:INTEGER);
PROCEDURE CHERCLASSE(CLASEXIST:EXICLASSE;NBRECLASSE:INTEGER;INDL:INTEGER;VAR
                     SORTIE:CLASSE);
PROCEDURE ECRIVERBE(LISTVERBE:LISTE;LISTMOT:TABCAR;INDV:INTEGER);
PROCEDURE CONT;
PROCEDURE ECRCLASU(CLASEXIST:EXICLASSE;NBRECLASSE:INTEGER;VAR COLONNE,LIGNE:
                   INTEGER);

IMPLEMENTATION

PROCEDURE CHARGCLASSUP;
VAR IND:INTEGER;
BEGIN
  IND:=0;
  (*$I-*)
  RESET(SUPPLECLASSES,'#5:CLASSESUP.DATA');
  IF IORESULT = 0
   THEN
    BEGIN
     (*$I+*)
     PASCREE:=FALSE;
     WHILE NOT EOF(SUPPLECLASSES)DO
      BEGIN
       IND:=IND+1;
       SUPPLEMENTAIRECLASSE[IND]:=SUPPLECLASSES^;
       GET(SUPPLECLASSES)
      END
    END
   ELSE PASCREE:=TRUE;
  CLOSE(SUPPLECLASSES,LOCK);
  DERNCLASSUP:=IND
END;(*CHARGCLASSUP*)

PROCEDURE LOADCLASSEMOT;
VAR IND:INTEGER;
BEGIN
  IND:=0;
  LISTNROCLASSE:=[];
  LINOCLASU:=[];
  LINOCLALI:=[];
  NBCLALIEU:=0;
  NBCLASUJET:=0;
  (*$I-*)
  RESET(CLASSEMOT,'#5:LISTCLAS.DATA');
  IF IORESULT = 0
   THEN
    BEGIN
     (*$I+*)
     PASCREE:=FALSE;
     WHILE NOT EOF(CLASSEMOT)DO
       BEGIN
         IND:=IND+1;
         CLASEXIST[IND]:=CLASSEMOT^;
```

```
          LISTNROCLASSE:=LISTNROCLASSE+[CLASEXIST[IND].NUMERO];
          IF CLASEXIST[IND].NUMERO<=100
            THEN
             BEGIN
              LINOCLASU:=LINOCLASU+[CLASEXIST[IND].NUMERO];
              NBCLASUJET:=NBCLASUJET+1
             END
            ELSE
             BEGIN
              LINOCLALI:=LINOCLALI+[CLASEXIST[IND].NUMERO];
              NBCLALIEU:=NBCLALIEU+1
             END;
          GET(CLASSEMOT)
        END
      END
    ELSE PASCREE:=TRUE;
  CLOSE(CLASSEMOT,LOCK);
  NBRECLASSE:=IND
END;

PROCEDURE CORRECTIONPHRASES;
VAR INDL,INDV,INDS:INTEGER;
BEGIN
  FOR INDS:=1 TO NBCLASUJET DO
   FOR INDV:=1 TO NBREVERBE DO
    FOR INDL:=0 TO NBCLALIEU DO
      CORRECTSEM[INDS,INDV,INDL]:=FALSE;
  (*$I-*)
  RESET(PHRASECORRECTE,'#5:PHRACOR.DATA');
  IF IORESULT = 0   (*$I+*)
   THEN
    BEGIN
     PASCREE:=FALSE;
     WHILE NOT EOF(PHRASECORRECTE)DO
       BEGIN
        WITH PHRASECORRECTE^ DO CORRECTSEM[SUJET,VERBE,LIEU]:=VALEUR;
        GET(PHRASECORRECTE)
       END
    END
  ELSE PASCREE:=TRUE;
  CLOSE(PHRASECORRECTE,LOCK)
END;(*CORRECTIONPHRASES*)

PROCEDURE CHARBYTE;
BEGIN
  LONGUEUR:=LONGUEUR+1;
  MOVELEFT(LISTMOT[SOURCE-1],DESTINATION,LONGUEUR);
  LONGUEUR:=LONGUEUR-1;
  MOVELEFT(LONGUEUR,DESTINATION,1)
END;

PROCEDURE RECHERCHE;
VAR BS,DEMI,INDCAR,LONGUEUR:INTEGER;
    INTER:CHAINE;
    CONTINUE:BOOLEAN;
BEGIN
  ORTHO:=FALSE;
  CONTINUE:=TRUE;
  BI:=1;
  BS:=NBREMOT-1;
  WHILE (CONTINUE)AND(BI<=BS) DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      INDCAR:=INDICE[DEMI].INDLIS;
      LONGUEUR:=INDICE[DEMI+1].INDLIS-INDCAR-1;
      CHARBYTE(LISTMOT,INDCAR,LONGUEUR,INTER);
      IF ENTREE=INTER
            THEN
```

```
                BEGIN
                 CONTINUE:=FALSE;
                 INDMOT:=DEMI;
                 ORTHO:=TRUE
                END
               ELSE IF ENTREE<INTER
                     THEN BS:=DEMI-1
                     ELSE BI:=DEMI+1
    END;
END;

PROCEDURE RECHSUVERLIEU;
VAR BS,DEMI:INTEGER;
    CONTINUE:BOOLEAN;
BEGIN
  CONTINUE:=TRUE;
  ORTHO:=FALSE;
  BI:=1;
  BS:=NBRE;
  WHILE CONTINUE AND(BI<=BS)DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      IF INDCARAP=ENS[DEMI].INDICE
        THEN
          BEGIN
            CONTINUE:=FALSE;
            INDSUVERLIEU:=DEMI;
            ORTHO:=TRUE
          END
        ELSE IF INDCARAP < ENS[DEMI].INDICE
               THEN BS:=DEMI-1
               ELSE BI:=DEMI+1
    END;
END;

PROCEDURE CONT;
VAR REP:CHAR;
BEGIN
  GOTOXY(0,23);
  WRITE(VIDE);
  GOTOXY(0,23);
  WRITE('TAPEZ <SPACE> POUR CONTINUER!');
  READ(KEYBOARD,REP);
  WHILE ORD(REP)<>32 DO
    BEGIN
      GOTOXY(27,23);
      READ(KEYBOARD,REP)
    END;
  GOTOXY(0,23);WRITE(VIDE)
END;

PROCEDURE CHERCLASSE;
VAR BI,BS,DEMI:INTEGER;
    TROUVE,CONTINUE:BOOLEAN;
BEGIN
  CONTINUE:=TRUE;
  BI:=1;
  BS:=NBRECLASSE;
  TROUVE:=FALSE;
  WHILE CONTINUE AND (BI<=BS)DO
    BEGIN
      DEMI:=(BI+BS)DIV 2;
      IF CLASEXIST[DEMI].NUMERO = INDL
        THEN
          BEGIN
            CONTINUE:=FALSE;
            TROUVE:=TRUE;
            SORTIE:=CLASEXIST[DEMI]
```

```
          END
        ELSE IF INDL < CLASEXIST[DEMI].NUMERO
          THEN BS:=DEMI-1
          ELSE BI:=DEMI+1
    END
END;(*CHERCLASSE*)

PROCEDURE ECRIVERBE;
VAR I:INTEGER;
BEGIN
 WRITE('VERBE:',INDV,':');
 I:=LISTVERBE[INDV].INDICE;
 WHILE LISTMOT[I] <>'@' DO
  BEGIN
   WRITE(LISTMOT[I]);
   I:=I+1
  END;
 WRITE(VIDE)
END;(*ECRIVERBE*)

PROCEDURE ECRCLASU;
VAR I:INTEGER;
BEGIN
 I:=1;
 WHILE(CLASEXIST[I].NUMERO<51)AND(I<=NBRECLASSE)DO
   BEGIN
     GOTOXY(COLONNE,LIGNE);
     WRITE(CLASEXIST[I].NUMERO,' ',CLASEXIST[I].TITRE);
     I:=I+1;
     IF LIGNE<18 THEN LIGNE:=LIGNE+1
                 ELSE
                   BEGIN
                     LIGNE:=12;
                     COLONNE:=25
                   END
   END
END;

BEGIN
END.
```

```
(*$S+*)

UNIT U2;

INTERFACE

USES(*$U APPLE1:U1.CODE*) U1;

VAR DICTIONNAIRE:FILE OF TYPMOT;

PROCEDURE ECRCLALI(CLASEXIST:EXICLASSE;NBRECLASSE:INTEGER;VAR COLONNE,LIGNE:
                   INTEGER);
PROCEDURE ECRCLASSEX(CLASEXIST:EXICLASSE;NBRECLASSE,LIMITE:INTEGER;VAR COLONNE
                     ,LIGNE:INTEGER);
PROCEDURE SAUVEPHRASESCORRECTES(CORRECTSEM:CORRECT;NBCLASUJET,NBREVERBE,
                                NBCLALIEU:INTEGER);
PROCEDURE SAUVECLASSUP(SUPPLEMENTAIRECLASSE:LICLASSUP;DERNCLASSUP:INTEGER);

PROCEDURE SAVECLASSEMOT(CLASEXIST:EXICLASSE;NBRECLASSE:INTEGER);
PROCEDURE MENUGENERAL(VAR CHOIX:CHAR);
PROCEDURE GESMOTS(VAR CHOIXMOT:CHAR);
PROCEDURE GESPHRASES(VAR CHOIXPHRASES:CHAR);
PROCEDURE CLASSEXIST;
PROCEDURE IMPRESCLAS;
PROCEDURE CONTARRET(LIGNE,NBRE:INTEGER;VAR FINI:BOOLEAN);
PROCEDURE VERBECARAC(LISTVERBE:LISTE;INDVERBE:INTEGER;VAR LIGNE:INTEGER);
PROCEDURE MOTCARAC(ENS:LISTE;DIM:INTEGER;VAR INDMOT,LIGNE:INTEGER);
PROCEDURE SULIEUCARAC(LISTLIEU:LISTE;NBRELIEU:INTEGER;VAR INDL:INTEGER);
PROCEDURE CONVERSION(CARAC:CHAINE;VAR ENTIER:INTEGER);
PROCEDURE VERBEX(LISTVERBE:LISTE;LISTMOT:TABCAR;NBREVERBE:INTEGER;VAR COLONNE,
                 LIGNE:INTEGER);
PROCEDURE CHARGVERBE(VAR LISTMOT:TABCAR;VAR INDICE:TABIND;VAR LISTVERBE:LISTE;
                     VAR NBREMOT,NBREVERBE:INTEGER;VAR PASCREE:BOOLEAN);
PROCEDURE TERMINER;


IMPLEMENTATION

PROCEDURE ECRCLALI;
VAR I:INTEGER;
BEGIN
 FOR I:=1 TO NBRECLASSE DO
   BEGIN
     IF CLASEXIST[I].NUMERO>100 THEN
       BEGIN
         GOTOXY(COLONNE,LIGNE);
         WRITE(CLASEXIST[I].NUMERO,' ',CLASEXIST[I].TITRE);
         IF LIGNE < 18 THEN LIGNE:=LIGNE+1
                       ELSE
                         BEGIN
                           LIGNE:=12;
                           COLONNE:=25
                         END
       END
   END
END;

PROCEDURE ECRCLASSEX;
VAR I,DEBUT:INTEGER;

 PROCEDURE ALALIGNE;
 BEGIN
  IF LIGNE>=LIMITE THEN
                    BEGIN
                      CONT;
                      LIGNE:=DEBUT;
                      IF COLONNE<20 THEN COLONNE:=20
                                    ELSE COLONNE:=42
```

```
                    END
                  ELSE LIGNE:=LIGNE+1
 END;
BEGIN
  GOTOXY(COLONNE,LIGNE);WRITE('LISTE DES CLASSES DE MOTS EXISTANTES');
  GOTOXY(COLONNE-1,LIGNE+1);FOR I:=1 TO 38 DO WRITE('-');
  LIGNE:=LIGNE+2;
  DEBUT:=LIGNE;
  COLONNE:=0;
  FOR I:=1 TO NBRECLASSE DO
    BEGIN
      GOTOXY(COLONNE,LIGNE);
      WRITE(VIDE);
      GOTOXY(COLONNE,LIGNE);
      WRITE(CLASEXIST[I].NUMERO,' ',CLASEXIST[I].TITRE);
      ALALIGNE;
    END;
  GOTOXY(COLONNE,LIGNE);
  WRITE('201:PREP "EN"');
  ALALIGNE;
  GOTOXY(COLONNE,LIGNE);
  WRITE('301:PREP "A"');
  ALALIGNE;
  GOTOXY(COLONNE,LIGNE);
  WRITE('401:PREP "A"OU"EN"');
  GOTOXY(0,22);WRITE('FIN DE LA LISTE!')
END;

PROCEDURE SAUVEPHRASECORRECTE;
VAR INDS,INDV,INDL:INTEGER;
BEGIN
 REWRITE(PHRASECORRECTE,'#5:PHRACOR.DATA');
 FOR INDS:=1 TO NBCLASUJET DO
  FOR INDV:=1 TO NBREVERBE DO
   FOR INDL:=0 TO NBCLALIEU DO
    BEGIN
     IF CORRECTSEM[INDS,INDV,INDL] THEN
      WITH PHRASECORRECTE^ DO
       BEGIN
        VALEUR:=CORRECTSEM[INDS,INDV,INDL];
        SUJET:=INDS;
        VERBE:=INDV;
        LIEU:=INDL
       END;
     PUT(PHRASECORRECTE)
    END;
 CLOSE(PHRASECORRECTE,LOCK)
END;(*SAUVEPHRASECORRECTE*)

PROCEDURE SAUVECLASSUP;
VAR IND:INTEGER;
BEGIN
  REWRITE(SUPPLECLASSES,'#5:CLASSESUP.DATA');
  FOR IND:=1 TO DERNCLASSUP DO
    BEGIN
      SUPPLECLASSES^:=SUPPLEMENTAIRECLASSE[IND];
      PUT(SUPPLECLASSES)
    END;
  CLOSE(SUPPLECLASSES,LOCK);
END;(*SAUVECLASSUP*)

PROCEDURE SAVECLASSEMOT;
VAR IND:INTEGER;
BEGIN
  REWRITE(CLASSEMOT,'#5:LISTCLAS.DATA');
  FOR IND:=1 TO NBRECLASSE DO
    BEGIN
      CLASSEMOT^:=CLASEXIST[IND];
```

```
      PUT(CLASSEMOT)
    END;
  CLOSE(CLASSEMOT,LOCK)
END;(*SAVECLASSEMOT*)

PROCEDURE MENUGENERAL;
VAR I:INTEGER;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(15,0);WRITE('MENU GENERAL');
  GOTOXY(14,1);FOR I:=1 TO 14 DO WRITE('=');
  GOTOXY(4,2);WRITE('1:GESTION DES MOTS ET DE LEURS');
  GOTOXY(8,3);WRITE('PROPRIETES:ORTHOGRAPHE-GENRE-');
  GOTOXY(8,4);WRITE('POSSIBILITE DE PLURIEL-CLASSE(S)');
  GOTOXY(4,7);WRITE('2:GESTION DES PHRASES GENEREES PAR');
  GOTOXY(8,8);WRITE('LE SYSTEME:');
  GOTOXY(9,9);WRITE('-CLASSES SUJETS:DE 1 A 100');
  GOTOXY(12,10);WRITE('-CLASSES LIEUX:DE 101 A 200');
  GOTOXY(12,11);WRITE('-CLASSES VERBES:201,301 ET 401');
  GOTOXY(5,12);WRITE('*POUR UNE CLASSE SUJET ET UN VERBE:');
  GOTOXY(7,13);WRITE('VERBES QUI ACCEPTENT CETTE CLASSE');
  GOTOXY(5,14);WRITE('*POUR UNE CLASSE SUJET ET UN VERBE:');
  GOTOXY(7,15);WRITE('TOUTES LES CLASSES LIEU QUI CON-');
  GOTOXY(7,16);WRITE('VIENNENT.');
  GOTOXY(5,19);WRITE('3:FIN DU TRAVAIL');
  GOTOXY(25,22);WRITE('VOTRE CHOIX:');
  READ(KEYBOARD,CHOIX);
  WHILE(CHOIX<>'1')AND(CHOIX<>'2')AND(CHOIX<>'3')DO
    BEGIN
      GOTOXY(2,20);WRITE('ATTENTION!');
      GOTOXY(2,21);WRITE('TAPEZ SEULEMENT 1,2 OU 3!');
      GOTOXY(37,22);READ(KEYBOARD,CHOIX)
    END
END;(*MENU GENERAL*)

PROCEDURE GESMOTS;
VAR I:INTEGER;
    ACCEPTE:SET8;
BEGIN
  PAGE(OUTPUT);
  ACCEPTE:=['1'..'8'];
  GOTOXY(11,0);WRITE('MENU GESTION DE MOTS');
  GOTOXY(10,1);FOR I:=1 TO 21 DO WRITE('=');
  GOTOXY(3,4);WRITE('1:CONSULTATION DU DICTIONNAIRE');
  GOTOXY(3,6);WRITE('2:AJOUT DE MOT(S)');
  GOTOXY(3,8);WRITE('3:MODIFICATION DE MOT(S)');
  GOTOXY(3,10);WRITE('4:SUPPRESSION DE MOT(S)');
  GOTOXY(3,12);WRITE('5:CONSULTATION DE MOT(S)');
  GOTOXY(3,14);WRITE('6:CREATION DE CLASSE(S) DE MOTS');
  GOTOXY(3,16);WRITE('7:RETOUR AU MENU GENERAL');
  GOTOXY(3,18);WRITE('8:FIN DU TRAVAIL');
  GOTOXY(22,23);WRITE('VOTRE CHOIX:');
  READ(KEYBOARD,CHOIXMOT);
  WHILE NOT(CHOIXMOT IN ACCEPTE) DO
    BEGIN
      GOTOXY(3,21);WRITE('ATTENTION!');
      GOTOXY(3,22);WRITE('TAPEZ SEULEMENT UN NOMBRE COMPRIS ');
      GOTOXY(4,23);WRITE('ENTRE 1 ET 8');
      GOTOXY(34,23);READ(KEYBOARD,CHOIXMOT)
    END
END;(*GESMOTS*)

PROCEDURE GESPHRASES;
VAR I:INTEGER;
    ACCEPTE:SET6;
BEGIN
  PAGE(OUTPUT);
  ACCEPTE:=['1'..'5'];
```

```
  GOTOXY(9,0);WRITE('MENU GESTION DE PHRASES');
  GOTOXY(8,1);FOR I:=1 TO 25 DO WRITE('=');
  GOTOXY(0,4);WRITE('1:CONSULTATION DE LA LISTE DES PHRASES ');
  GOTOXY(3,5);WRITE('CORRECTES');
  GOTOXY(0,8);WRITE('2:AJOUT DE PHRASE(S)');
  GOTOXY(0,11);WRITE('3:SUPPRESSION DE PHRASE(S)');
  GOTOXY(0,14);WRITE('4:RETOUR AU MENU GENERAL');
  GOTOXY(0,17);WRITE('5:FIN DU TRAVAIL');
  GOTOXY(22,23);WRITE('VOTRE CHOIX:');
  READ(KEYBOARD,CHOIXPHRASES);
  WHILE NOT(CHOIXPHRASES IN ACCEPTE) DO
    BEGIN
      GOTOXY(3,21);WRITE('ATTENTION!');
      GOTOXY(3,22);WRITE('TAPEZ SEULEMENT UN NOMBRE COMPRIS ');
      GOTOXY(4,23);WRITE('ENTRE 1 ET 5');
      GOTOXY(34,23);READ(KEYBOARD,CHOIXPHRASES)
    END
END;(*GESPHRASES*)

PROCEDURE CLASSEXIST;
VAR LIGNE,COLONNE,NBRECLASSE,NBCLASUJET,NBCLALIEU:INTEGER;
    PASCREE:BOOLEAN;
    REP:CHAR;
    CLASEXIST:EXICLASSE;
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;
BEGIN
  PAGE(OUTPUT);
  LIGNE:=0;
  COLONNE:=1;
  LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
                LINOCLASU,LINOCLALI,PASCREE);
  IF NOT PASCREE
   THEN ECRCLASSEX(CLASEXIST,NBRECLASSE,21,COLONNE,LIGNE)
   ELSE
    BEGIN
     GOTOXY(COLONNE,LIGNE);
     WRITE('LISTE DES CLASSES DE MOTS EXISTANTES');
     GOTOXY(COLONNE-1,LIGNE+1);
     FOR COLONNE:=1 TO 38 DO WRITE('-');
     GOTOXY(0,10);
     WRITE('IL N''Y A PAS DE CLASSES DE MOTS')
    END;
  CONT
END;(*CLASSEXIST*)

PROCEDURE IMPRESCLAS;
VAR I,COLONNE,NBRECLASSE,NBCLASUJET,NBCLALIEU:INTEGER;
    TITRE,STRNRO,INTITULE:CHAINE;
    PASCLAS:BOOLEAN;
    CRRET,REP,BLANC:CHAR;
    CLASEXIST:EXICLASSE;
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;

 PROCEDURE IMPRESELEM(STRNRO,INTITULE:CHAINE);
 BEGIN
   UNITWRITE(6,TITRE[1],LENGTH(TITRE));
   UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
   FOR COLONNE:=1 TO 3 DO UNITWRITE(6,BLANC,1);
   UNITWRITE(6,INTITULE[1],LENGTH(INTITULE));
   UNITWRITE(6,CRRET,1)
 END;

BEGIN
  PAGE(OUTPUT);
```

```
  GOTOXY(2,6);WRITE('IMPRESSION DE LA LISTE DES CLASSES');
  GOTOXY(11,8);WRITE('DE MOTS EXISTANTES');
  GOTOXY(3,11);WRITE('VERIFIEZ SI L''IMPRIMANTE EST BRANCHEE!');
  GOTOXY(3,13);WRITE('PUIS TAPEZ N''IMPORTE QUELLE TOUCHE!');
  READ(KEYBOARD,REP);
  (*$I-*)
  UNITCLEAR(6);
  (*$I+*)
  IF IORESULT<>0
    THEN
      BEGIN
        GOTOXY(3,15);
        WRITE('L''IMPRIMANTE N''EST PAS BRANCHEE!')
      END
    ELSE
      BEGIN
        TITRE:='CLASSE:';
        CRRET:=CHR(13);
        BLANC:=' ';
        LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
                      LINOCLASU,LINOCLALI,PASCLAS);
        IF NOT PASCLAS
         THEN
          BEGIN
           GOTOXY(0,11);FOR I:=1 TO 4 DO WRITELN(VIDE);
           GOTOXY(0,11);WRITE('IMPRESSION EN COURS');
           FOR I:=1 TO NBRECLASSE DO
             BEGIN
               STR(CLASEXIST[I].NUMERO,STRNRO);
               INTITULE:=CLASEXIST[I].TITRE;
               IMPRESELEM(STRNRO,INTITULE)
             END;
           IMPRESELEM('201','PREPOSITION "EN"');
           IMPRESELEM('301','PREPOSITION "A"');
           IMPRESELEM('401','PREPOSITION "A" OU "EN"');
           GOTOXY(0,11);WRITE(VIDE);
           GOTOXY(0,11);WRITE('IMPRESSION TERMINEE')
          END
         ELSE
          BEGIN
           GOTOXY(0,11);
           FOR I:=1 TO 2 DO WRITELN(VIDE);
           GOTOXY(0,11);
           WRITE('IL N''Y A PAS DE CLASSES EXISTANTES')
          END
      END;
  CONT
END;(*IMPRESCLAS*)

PROCEDURE CONTARRET;
VAR REP:CHAR;
    I:INTEGER;
BEGIN
 GOTOXY(0,23);WRITE('POUR CONTINUER,TAPEZ"C",POUR ARRETER:"A"');
 READ(KEYBOARD,REP);
 WHILE (REP<>'A')AND(REP<>'C')DO
  BEGIN
   GOTOXY(40,23);
   READ(KEYBOARD,REP)
  END;
 IF REP='A'
  THEN FINI:=TRUE
  ELSE
   BEGIN
    GOTOXY(0,LIGNE);
    FINI:=FALSE;
    FOR I:=LIGNE TO NBRE DO
     BEGIN
```

```
      GOTOXY(0,I);
      WRITE(VIDE)
     END
   END;
  GOTOXY(0,23);WRITE(VIDE)
END;(*CONTARRET*)

PROCEDURE VERBECARAC;
VAR I:INTEGER;
BEGIN
 LIGNE:=LIGNE+1;
 GOTOXY(0,LIGNE);FOR I:=1 TO 2 DO WRITELN(VIDE);
 GOTOXY(2,LIGNE);
 WRITE('CLASSE ',LISTVERBE[INDVERBE].CLASSE,' ');
 WRITE('PREPOSITION DEMANDEE:');
 CASE LISTVERBE[INDVERBE].CLASSE OF
   201:WRITE('"EN"');
   301:WRITE('"A"');
   401:WRITE('"EN-A"')
  END;(*CASE*)
 LIGNE:=LIGNE+1
END;(*VERBECARAC*)

PROCEDURE MOTCARAC;
VAR MOTAP,NROLIGNE,I:INTEGER;
BEGIN
  NROLIGNE:=LIGNE+1;
  GOTOXY(0,NROLIGNE);FOR I:=1 TO 5 DO WRITELN(VIDE);
  GOTOXY(2,NROLIGNE);WRITE('GENRE(M/F):');
  IF ENS[INDMOT].GENRE THEN WRITE('MASCULIN')
                       ELSE WRITE('FEMININ');
  NROLIGNE:=NROLIGNE+1;
  GOTOXY(2,NROLIGNE);
  WRITE('POSSIBILITE DE PLURIEL(O/N):');
  IF ENS[INDMOT].POSPLUR THEN WRITE('OUI')
                         ELSE WRITE('NON');
  NROLIGNE:=NROLIGNE+1;
  GOTOXY(2,NROLIGNE);
  WRITE('POSSIBILITE DE DETERMINANT(O/N):');
  IF ENS[INDMOT].POSDET THEN WRITE('OUI')
                        ELSE WRITE('NON');
  NROLIGNE:=NROLIGNE+1;
  GOTOXY(2,NROLIGNE);
  WRITE('CLASSE:',ENS[INDMOT].CLASSE);
  MOTAP:=INDMOT-1;
  WHILE MOTAP>0 DO
   IF ENS[INDMOT].INDICE=ENS[MOTAP].INDICE
     THEN
       BEGIN
        WRITE(' ',ENS[MOTAP].CLASSE);
        MOTAP:=MOTAP-1
       END
     ELSE MOTAP:=0;
  MOTAP:=INDMOT+1;
  WHILE MOTAP<=DIM DO
   IF ENS[INDMOT].INDICE=ENS[MOTAP].INDICE
    THEN
     BEGIN
      WRITE(' ',ENS[MOTAP].CLASSE);
      INDMOT:=MOTAP;
      MOTAP:=MOTAP+1
     END
    ELSE MOTAP:=DIM+1;
  INDMOT:=INDMOT+1;
  LIGNE:=NROLIGNE+1
END;(*MOTCARAC*)

PROCEDURE SULIEUCARAC;
```

```
VAR MOTTAP:INTEGER;
BEGIN
 MOTTAP:=INDL;
 WHILE MOTTAP>0 DO
  IF LISTLIEU[INDL].INDICE=LISTLIEU[MOTTAP].INDICE
   THEN
    BEGIN
     WRITE(' ',LISTLIEU[MOTTAP].CLASSE);
     MOTTAP:=MOTTAP-1
    END
   ELSE MOTTAP:=0;
 MOTTAP:=INDL+1;
 WHILE MOTTAP<=NBRELIEU DO
  IF LISTLIEU[MOTTAP].INDICE=LISTLIEU[INDL].INDICE
   THEN
    BEGIN
     WRITE(' ',LISTLIEU[MOTTAP].CLASSE);
     INDL:=MOTTAP;
     MOTTAP:=MOTTAP+1
    END
   ELSE MOTTAP:=NBRELIEU+1;
 INDL:=INDL+1
END;(*SULIEUCARAC*)

PROCEDURE CONVERSION;
VAR CHIFFRE:SET OF '0'..'9';
    STOP:BOOLEAN;
    INTER,I,PLACE:INTEGER;
BEGIN
  ENTIER:=0;
  CHIFFRE:=['0','1','2','3','4','5','6','7','8','9'];
  IF LENGTH(CARAC)>0 THEN
    BEGIN
      PLACE:=1;
      STOP:=FALSE;
      I:=LENGTH(CARAC);
      WHILE (CARAC[I] IN CHIFFRE) AND (NOT STOP) DO
        BEGIN
          CASE CARAC[I] OF
            '1':INTER:=1;
            '2':INTER:=2;
            '3':INTER:=3;
            '4':INTER:=4;
            '5':INTER:=5;
            '6':INTER:=6;
            '7':INTER:=7;
            '8':INTER:=8;
            '9':INTER:=9;
            '0':INTER:=0
           END;(*CASE*)
          ENTIER:=ENTIER+(INTER*PLACE);
          IF I=1 THEN STOP:=TRUE
                 ELSE
                   BEGIN
                     I:=I-1;
                     PLACE:=PLACE*10
                   END
        END;
      IF NOT STOP THEN ENTIER:=0
    END
END;(*CONVERSION*)

PROCEDURE VERBEX;
VAR DEPART,I:INTEGER;
BEGIN
 DEPART:=LIGNE;
 FOR I:=1 TO NBREVERBE DO
  BEGIN
```

```
    GOTOXY(COLONNE,LIGNE);
    ECRIVERBE(LISTVERBE,LISTMOT,I);
    IF LIGNE<18 THEN LIGNE:=LIGNE+1
                ELSE
                 BEGIN
                  LIGNE:=DEPART;
                  COLONNE:=25
                 END
  END
END;(*VERBEX*)

PROCEDURE CHARGVERBE;
VAR IND,INDV:INTEGER;
    ENTREE:TYPMOT;
BEGIN
  INDV:=0;
  IND:=1;
  LISTMOT[IND]:='@';
  (*$I-*)
  RESET(DICTIONNAIRE,'#5:DICTIO.DATA');
  IF IORESULT = 0
   THEN   (*$I+*)
    BEGIN
      PASCREE:=FALSE;
      WHILE NOT EOF(DICTIONNAIRE)DO
        BEGIN
          ENTREE:=DICTIONNAIRE^;
          IF ENTREE.CLASSE>199 THEN
            BEGIN
              INDV:=INDV+1;
              IND:=IND+1;
              INDICE[INDV].INDLIS:=IND;
              INDICE[INDV].PTRTYP:='V';
              LISTVERBE[INDV].INDICE:=IND;
              LISTVERBE[INDV].CLASSE:=ENTREE.CLASSE;
              MOVELEFT(ENTREE.MOT[1],LISTMOT[IND],LENGTH(ENTREE.MOT));
              IND:=IND+LENGTH(ENTREE.MOT);
              LISTMOT[IND]:='@'
            END;
          GET(DICTIONNAIRE)
        END;(*WHILE NOT EOF*)
      CLOSE(DICTIONNAIRE,LOCK)
    END(*IORESULT*)
   ELSE PASCREE:=TRUE;
 NBREVERBE:=INDV;
 INDICE[INDV+1].INDLIS:=IND+1;
 LISTMOT[IND+1]:='Z';
 LISTMOT[IND+2]:='Z';
 LISTMOT[IND+3]:='Z';
 NBREMOT:=INDV+1
END;(*CHARGVERBE*)

PROCEDURE TERMINER;
VAR LIGNE:INTEGER;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(4,7);
  WRITE('FIN DE LA GESTION DU DICTIONNAIRE');
  GOTOXY(16,14);
  WRITE('AU REVOIR!')
END;

BEGIN
END.
```

```
(*$S+*)
UNIT U3;

INTERFACE

USES (*$U APPLE1:U1.CODE*)U1;

PROCEDURE ACCUEIL;

IMPLEMENTATION

PROCEDURE ACCUEIL;

PROCEDURE ACCUEIL1;
VAR I:INTEGER;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(9,0);WRITE('GESTION DU DICTIONNAIRE');
  GOTOXY(8,1);FOR I:=1 TO 26 DO WRITE('=');
  GOTOXY(5,2);WRITE('BONJOUR!');
  GOTOXY(5,3);WRITE('CECI EST UN SYSTEME DE GESTION DU');
  GOTOXY(0,4);WRITE('DICTIONNAIRE EMPLOYE PAR LE SYSTEME DE');
  GOTOXY(0,5);WRITE('GENERATION DE PHRASES DESTINEES AUX');
  GOTOXY(0,6);WRITE('EXERCICES SUR L''EMPLOI DES PREPOSITIONS');
  GOTOXY(0,7);WRITE('DE LIEU "A" ET "EN" EN ESPAGNOL');
  GOTOXY(5,8);WRITE('LES MOTS QUI PEUVENT ETRE DES NOMS,');
  GOTOXY(0,9);WRITE('DES PRONOMS PERSONNELS OU DES VERBES,');
  GOTOXY(0,10);WRITE('SONT REGROUPES EN CLASSES HOMOGENES');
  GOTOXY(0,11);WRITE('POUR LES VERBES,3 CLASSES SONT FIXEES:');
  GOTOXY(3,12);WRITE('-CLASSE 201:VERBES AVEC LA PREP "EN"');
  GOTOXY(3,13);WRITE('-CLASSE 301:VERBES AVEC LA PREP "A"');
  GOTOXY(3,14);WRITE('-CLASSE 401:VERBES AVEC LES DEUX PREP');
  GOTOXY(5,15);WRITE('LES NOMS ET LES PRONOMS PERSONNELS');
  GOTOXY(2,16);WRITE('PEUVENT APPARTENIR A PLUSIEURS CLASSES');
  GOTOXY(2,17);WRITE('SUJETS ET/OU LIEUX DIFFERENTES;ILS ');
  GOTOXY(2,18);WRITE('AURONT DANS CE CAS PLUS DE CHANCE D" ');
  GOTOXY(2,19);WRITE('ETRE GENERES CAR ILS SERONT REPRIS ');
  GOTOXY(2,20);WRITE('PLUSIEURS FOIS DANS LES LISTES DE ');
  GOTOXY(2,21);WRITE('MOTS.CE N"EST PAS LE CAS POUR LES VERBES')
END;(*ACCUEIL1*)

PROCEDURE ACCUEIL2;
BEGIN
  PAGE(OUTPUT);
  GOTOXY(0,0);WRITE('GESTION DU DICTIONNAIRE(SUITE)');
  GOTOXY(5,2);WRITE('LE SYSTEME GENERE ALEATOIREMENT');
  GOTOXY(0,3);WRITE('UN SUJET,PUIS UN VERBE QUI CONVIENNE A');
  GOTOXY(0,4);WRITE('LA CLASSE DONT LE SUJET FAIT PARTIE ET');
  GOTOXY(0,5);WRITE('ENFIN UN COMPLEMENT DE LIEU QUI APPAR-');
  GOTOXY(0,6);WRITE('TIENNE A UNE CLASSE ACCEPTEE PAR LA');
  GOTOXY(0,7);WRITE('CLASSE SUJET ET LE VERBE DEJA GENERES.');
  GOTOXY(5,8);WRITE('UN MOT DU DICTIONNAIRE A PLUSIEURS');
  GOTOXY(0,9);WRITE('PROPRIETES:*SON ORTHOGRAPHE(L''ECRITURE');
  GOTOXY(13,10);WRITE('DU MOT SEUL ET EN ENTIER)');
  GOTOXY(12,11);WRITE('*SON GENRE:MASCULIN OU');
  GOTOXY(13,12);WRITE('FEMININ.');
  GOTOXY(12,13);WRITE('*SA POSSIBILITE D''ETRE MIS ');
  GOTOXY(13,14);WRITE('AU PLURIEL:EMPLOI AU PLURIEL');
  GOTOXY(12,15);WRITE('*LE NUMERO DE SA(SES)CLASSE(S)');
  GOTOXY(0,16);WRITE('N.B.:POUR UN VERBE,LE GENRE ET LA POS-');
  GOTOXY(6,17);WRITE('SIBILITE D''ETRE MIS AU PLURIEL N''');
  GOTOXY(6,18);WRITE('ONT AUCUNE IMPORTANCE.');
  GOTOXY(5,20);WRITE('LA GESTION DU DICTIONNAIRE SE SUB-');
  GOTOXY(0,21);WRITE('DIVISE EN:-GESTION DES MOTS');
  GOTOXY(10,22);WRITE('-GESTION DES PHRASES')
END;(*ACCUEIL2*)
```

```
BEGIN
  ACCUEIL1;
  CONT;
  ACCUEIL2;
  CONT
END;(*ACCUEIL*)

BEGIN
END.
```

```
(*$S+*)
UNIT U4;

INTERFACE

USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2;

PROCEDURE CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE:LICLASSUP;
        DERNCLASSUP:INTEGER;VAR LISTMOT:TABCAR;VAR LISTSUJET,LISTLIEU,LISTVERBE:
        LISTE;VAR INDICE:TABIND;VAR NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,
        NBRELIEU:INTEGER;VAR PASCREE:BOOLEAN);

IMPLEMENTATION

PROCEDURE CHARGEMENTDICTIONNAIRE;
VAR IND,INDI,INDS,INDV,INDL,CLASSESUIVANTE:INTEGER;
    ENTREE:TYPMOT;

BEGIN
  INDS:=0;INDV:=0;INDL:=0;
  IND:=1;INDI:=0;
  LISTMOT[IND]:='@';
  (*$I-*)
  RESET(DICTIONNAIRE,'#5:DICTIO.DATA');
  IF IORESULT=0
   THEN
    BEGIN
     (*$I+*)
     PASCREE:=FALSE;
     WHILE NOT EOF(DICTIONNAIRE) DO
       BEGIN
         INDI:=INDI+1;
         IND:=IND+1;
         INDICE[INDI].INDLIS:=IND;
         ENTREE:=DICTIONNAIRE^;
         IF ENTREE.CLASSE<=100
          THEN
           BEGIN
             INDS:=INDS+1;
             INDICE[INDI].PTRTYP:='S';
             LISTSUJET[INDS].INDICE:=IND;
             LISTSUJET[INDS].GENRE:=ENTREE.GENRE;
             LISTSUJET[INDS].POSPLUR:=ENTREE.POSPLUR;
             LISTSUJET[INDS].POSDET:=ENTREE.POSDET;
             LISTSUJET[INDS].CLASSE:=ENTREE.CLASSE
           END
          ELSE IF ENTREE.CLASSE<=200
           THEN
            BEGIN
              INDL:=INDL+1;
              INDICE[INDI].PTRTYP:='L';
              LISTLIEU[INDL].INDICE:=IND;
              LISTLIEU[INDL].GENRE:=ENTREE.GENRE;
              LISTLIEU[INDL].POSPLUR:=ENTREE.POSPLUR;
              LISTLIEU[INDL].POSDET:=ENTREE.POSDET;
              LISTLIEU[INDL].CLASSE:=ENTREE.CLASSE
            END
           ELSE
            BEGIN
              INDV:=INDV+1;
              INDICE[INDI].PTRTYP:='V';
              LISTVERBE[INDV].INDICE:=IND;
              LISTVERBE[INDV].CLASSE:=ENTREE.CLASSE
            END;
         CLASSESUIVANTE:=ENTREE.PTRCLASSE;
         WHILE CLASSESUIVANTE>0 DO
           BEGIN
             IF SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO<=100
```

```
                THEN
                 BEGIN
                   INDS:=INDS+1;
                   IF INDICE[INDI].PTRTYP='L' THEN INDICE[INDI].PTRTYP:='D';
                   LISTSUJET[INDS].INDICE:=IND;
                   LISTSUJET[INDS].GENRE:=ENTREE.GENRE;
                   LISTSUJET[INDS].POSPLUR:=ENTREE.POSPLUR;
                   LISTSUJET[INDS].POSDET:=ENTREE.POSDET;
                   LISTSUJET[INDS].CLASSE:=
                       SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO
                 END
                ELSE
                 BEGIN
                   INDL:=INDL+1;
                   IF INDICE[INDI].PTRTYP='S' THEN INDICE[INDI].PTRTYP:='D';
                   LISTLIEU[INDL].INDICE:=IND;
                   LISTLIEU[INDL].GENRE:=ENTREE.GENRE;
                   LISTLIEU[INDL].POSPLUR:=ENTREE.POSPLUR;
                   LISTLIEU[INDL].POSDET:=ENTREE.POSDET;
                   LISTLIEU[INDL].CLASSE:=
                     SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO
                 END;
               CLASSESUIVANTE:=SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].CLASPTR
             END;
           MOVELEFT(ENTREE.MOT[1],LISTMOT[IND],LENGTH(ENTREE.MOT));
           IND:=IND+LENGTH(ENTREE.MOT);
           LISTMOT[IND]:='@';
           GET(DICTIONNAIRE)
         END;(*WHILE NOT EOF*)
       CLOSE(DICTIONNAIRE,LOCK)
     END(*IORESULT*)
    ELSE PASCREE:=TRUE;
  INDICE[INDI+1].INDLIS:=IND+1;
  LISTMOT[IND+1]:='Z';
  LISTMOT[IND+2]:='Z';
  LISTMOT[IND+3]:='@';
  NBREMOT:=INDI+1;
  NBRECAR:=IND+3;
  NBRESUJET:=INDS;
  NBREVERBE:=INDV;
  NBRELIEU:=INDL
END;(*CHARGEMENTDICTIONNAIRE*)

BEGIN
END.
```

```
(*$S+*)
UNIT U5;

INTERFACE

USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2;

PROCEDURE SAUVETAGE(LISTMOT:TABCAR;INDICE:TABIND;LISTSUJET,LISTLIEU,LISTVERBE:
    LISTE;NBRECAR,NBREMOT,NBRESUJET,NBREVERBE,NBRELIEU:INTEGER;
    VAR SUPPLEMENTAIRECLASSE:LICLASSUP;VAR DERNCLASSUP:INTEGER);

IMPLEMENTATION

PROCEDURE SAUVETAGE;

VAR INDCAR,INDS,INDV,INDL,LONGUEUR,IND:INTEGER;
    SORTIE:TYPMOT;

 PROCEDURE SORTVERBE(NBREVERBE:INTEGER;VAR INDV:INTEGER);
 BEGIN
   SORTIE.CLASSE:=LISTVERBE[INDV].CLASSE;
   IF INDV<NBREVERBE THEN INDV:=INDV+1;
 END;(*SORTVERBE*)

PROCEDURE SORTSUJET(NBRESUJET:INTEGER;VAR INDS:INTEGER);
 VAR FINISUJET,DEUXIEME:BOOLEAN;
 BEGIN
   WITH SORTIE DO
    BEGIN
     GENRE:=LISTSUJET[INDS].GENRE;
     POSPLUR:=LISTSUJET[INDS].POSPLUR;
     POSDET:=LISTSUJET[INDS].POSDET;
     CLASSE:=LISTSUJET[INDS].CLASSE
    END;
   IF INDS<NBRESUJET
     THEN
       BEGIN
         DEUXIEME:=TRUE;
         FINISUJET:=FALSE;
         INDS:=INDS+1
       END
     ELSE FINISUJET:=TRUE;
   SORTIE.PTRCLASSE:=0;
   WHILE (LISTSUJET[INDS].INDICE=INDCAR) AND NOT FINISUJET DO
     BEGIN
       DERNCLASSUP:=DERNCLASSUP+1;
       IF DEUXIEME
         THEN
           BEGIN
             SORTIE.PTRCLASSE:=DERNCLASSUP;
             DEUXIEME:=FALSE
           END
         ELSE SUPPLEMENTAIRECLASSE[DERNCLASSUP-1].CLASPTR:=DERNCLASSUP;
       SUPPLEMENTAIRECLASSE[DERNCLASSUP].NUMERO:=LISTSUJET[INDS].CLASSE;
       IF INDS<NBRESUJET THEN
                          BEGIN
                           FINISUJET:=FALSE;
                           INDS:=INDS+1
                          END
                        ELSE FINISUJET:=TRUE
    END;(*WHILE*)
 END;(*SORTSUJET*)

 PROCEDURE SORTSULIEU(NBRESUJET,NBRELIEU:INTEGER;VAR INDS,INDL:INTEGER);
 VAR FINISUJET,FINILIEU,DEUXIEME:BOOLEAN;
 BEGIN
   WITH SORTIE DO
    BEGIN
```

```
    GENRE:=LISTSUJET[INDS].GENRE;
    POSPLUR:=LISTSUJET[INDS].POSPLUR;
    POSDET:=LISTSUJET[INDS].POSDET;
    CLASSE:=LISTSUJET[INDS].CLASSE
   END;
  IF INDS<NBRESUJET THEN
                      BEGIN
                        FINISUJET:=FALSE;
                        INDS:=INDS+1
                      END
                    ELSE FINISUJET:=TRUE;
  SORTIE.PTRCLASSE:=0;
  DEUXIEME:=TRUE;
  WHILE (LISTSUJET[INDS].INDICE=INDCAR) AND NOT FINISUJET DO
    BEGIN
      DERNCLASSUP:=DERNCLASSUP+1;
      IF DEUXIEME
        THEN
          BEGIN
            SORTIE.PTRCLASSE:=DERNCLASSUP;
            DEUXIEME:=FALSE
          END
        ELSE SUPPLEMENTAIRECLASSE[DERNCLASSUP-1].CLASPTR:=DERNCLASSUP;
      SUPPLEMENTAIRECLASSE[DERNCLASSUP].NUMERO:=LISTSUJET[INDS].CLASSE;
      IF INDS<NBRESUJET THEN
                          BEGIN
                            FINISUJET:=FALSE;
                            INDS:=INDS+1
                          END
                        ELSE FINISUJET:=TRUE
    END;(*WHILE*)
  IF INDL<=NBRELIEU THEN FINILIEU:=FALSE
                    ELSE FINILIEU:=TRUE;
  WHILE(LISTLIEU[INDL].INDICE=INDCAR)AND NOT FINILIEU DO
    BEGIN
      DERNCLASSUP:=DERNCLASSUP+1;
      IF DEUXIEME
        THEN
          BEGIN
            SORTIE.PTRCLASSE:=DERNCLASSUP;
            DEUXIEME:=FALSE
          END
        ELSE SUPPLEMENTAIRECLASSE[DERNCLASSUP-1].CLASPTR:=DERNCLASSUP;
      SUPPLEMENTAIRECLASSE[DERNCLASSUP].NUMERO:=LISTLIEU[INDL].CLASSE;
      IF INDL<NBRELIEU THEN
                         BEGIN
                           FINILIEU:=FALSE;
                           INDL:=INDL+1
                         END
                       ELSE FINILIEU:=TRUE
    END;(*WHILE*)
END;(*SORTSULIEU*)

PROCEDURE SORTLIEU(NBRELIEU:INTEGER;VAR INDL:INTEGER);
VAR FINILIEU,DEUXIEME:BOOLEAN;
BEGIN
  WITH SORTIE DO
   BEGIN
    GENRE:=LISTLIEU[INDL].GENRE;
    POSPLUR:=LISTLIEU[INDL].POSPLUR;
    POSDET:=LISTLIEU[INDL].POSDET;
    CLASSE:=LISTLIEU[INDL].CLASSE
   END;
  IF INDL<NBRELIEU
    THEN
      BEGIN
        DEUXIEME:=TRUE;
        FINILIEU:=FALSE;
```

```
          INDL:=INDL+1
        END
      ELSE FINILIEU:=TRUE;
    SORTIE.PTRCLASSE:=0;
    WHILE (LISTLIEU[INDL].INDICE=INDCAR)AND NOT FINILIEU DO
     BEGIN
        DERNCLASSUP:=DERNCLASSUP+1;
        IF DEUXIEME
          THEN
            BEGIN
              SORTIE.PTRCLASSE:=DERNCLASSUP;
              DEUXIEME:=FALSE
            END
          ELSE SUPPLEMENTAIRECLASSE[DERNCLASSUP-1].CLASPTR:=DERNCLASSUP;
        SUPPLEMENTAIRECLASSE[DERNCLASSUP].NUMERO:=LISTLIEU[INDL].CLASSE;
        IF INDL<NBRELIEU THEN
                              BEGIN
                                FINILIEU:=FALSE;
                                INDL:=INDL+1
                              END
                            ELSE FINILIEU:=TRUE
     END;(*WHILE*)
END;(*SORTLIEU*)

BEGIN
  REWRITE(DICTIONNAIRE,'#5:DICTIO.DATA');
  IF IORESULT = 0
   THEN
    BEGIN
     INDS:=1;INDV:=1;INDL:=1;DERNCLASSUP:=0;
     FOR IND:=1 TO NBREMOT-1 DO
       BEGIN
         INDCAR:=INDICE[IND].INDLIS;
         LONGUEUR:=INDICE[IND+1].INDLIS-INDCAR-1;
         CHARBYTE(LISTMOT,INDCAR,LONGUEUR,SORTIE.MOT);
         CASE INDICE[IND].PTRTYP OF
          'S':SORTSUJET(NBRESUJET,INDS);
          'V':SORTVERBE(NBREVERBE,INDV);
          'L':SORTLIEU(NBRELIEU,INDL);
          'D':SORTSULIEU(NBRESUJET,NBRELIEU,INDS,INDL)
          END;
         DICTIONNAIRE^:=SORTIE;
         PUT(DICTIONNAIRE)
       END;(*FOR*)
      CLOSE(DICTIONNAIRE,LOCK)
    END
   ELSE
    BEGIN
     GOTOXY(0,22);WRITE(VIDE);GOTOXY(0,22);WRITE('IMPOSSIBLE D''OUVRIR');
     WRITE('LE FICHIER DICTIONNAIRE');
     CONT
    END
END;(*SAUVETAGE*)

BEGIN
END.
```

```
(*$S+*)
UNIT U6;

INTERFACE

USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2,(*$U APPLE1:U4.CODE*)U4;

PROCEDURE LISTING;

IMPLEMENTATION

PROCEDURE LISTING;
VAR REP:CHAR;
    IND,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,I,DERNCLASSUP:INTEGER;
    ACCEPTE:SET OF '1'..'5';
    SUPPLEMENTAIRECLASSE:LICLASSUP;
    LISTMOT:TABCAR;
    LISTSUJET,LISTLIEU,LISTVERBE:LISTE;
    PASCREE:BOOLEAN;
    INDICE:TABIND;

PROCEDURE LISTECRAN;
VAR LONGUEUR,LIGNE,NRO,INDCAR,INDS,INDV,INDL,I:INTEGER;
    SORTIE:CHAINE;
    REP:CHAR;
BEGIN
  PAGE(OUTPUT);
  NRO:=0;
  GOTOXY(7,0);WRITE('LISTE ALPHABETIQUE DES MOTS');
  GOTOXY(6,1);FOR I:=1 TO 29 DO WRITE('-');
  CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASCREE);
  CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE,DERNCLASSUP,LISTMOT,LISTSUJET,
    LISTLIEU,LISTVERBE,INDICE,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,
    PASCREE);
  IF NOT PASCREE
   THEN
    BEGIN
     LIGNE:=2;
     INDS:=1;INDV:=1;INDL:=1;
     FOR IND:=1 TO (NBREMOT-1) DO
      BEGIN
       INDCAR:=INDICE[IND].INDLIS;
       LONGUEUR:=INDICE[IND+1].INDLIS-INDCAR-1;
       CHARBYTE(LISTMOT,INDCAR,LONGUEUR,SORTIE);
       NRO:=NRO+1;
       IF LIGNE>18 THEN
                    BEGIN
                      LIGNE:=2;
                      CONT
                     END;
       GOTOXY(0,LIGNE);WRITE(VIDE);
       GOTOXY(0,LIGNE);WRITE ('MOT ',NRO,':',SORTIE);
       CASE INDICE[IND].PTRTYP OF
        'S':MOTCARAC(LISTSUJET,NBRESUJET,INDS,LIGNE);
        'V':BEGIN
              VERBECARAC(LISTVERBE,INDV,LIGNE);
              IF INDV<NBREVERBE THEN INDV:=INDV+1
            END;
        'L':MOTCARAC(LISTLIEU,NBRELIEU,INDL,LIGNE);
        'D':BEGIN
              MOTCARAC(LISTSUJET,NBRESUJET,INDS,LIGNE);
              SULIEUCARAC(LISTLIEU,NBRELIEU,INDL)
            END
       END(*CASE*)
      END;(*FOR*)
     GOTOXY(0,22);
     WRITE('FIN DE LA LISTE')
    END
```

```
    ELSE
      BEGIN
        GOTOXY(0,10);
        WRITE('PAS DE MOTS DANS LA LISTE.');
        GOTOXY(0,13);
        WRITE('CELLE-CI N''EXISTE PAS!')
      END;
  CONT
END;(*LISTECRAN*)

PROCEDURE LISTIMPRIM;
VAR NRO,I,SUIVANT,DERNCLASSUP:INTEGER;
    BLANC,POINTS,REP,CRRET:CHAR;
    SORTIE:TYPMOT;
    PASCREE:BOOLEAN;
    PLURPOS,DETPOS:STRING[30];
    GENRE,MASCULIN,FEMININ,OUI,NON,CLASSE,STRNRO,TITRE:CHAINE;
    SUPPLEMENTAIRECLASSE:LICLASSUP;

PROCEDURE CORIMPRIM;
  BEGIN
    TITRE:='MOT NRO';
    BLANC:=' ';
    POINTS:=':';
    CRRET:=CHR(13);
    GENRE:='GENRE:';
    MASCULIN:='MASCULIN';
    FEMININ:='FEMININ';
    PLURPOS:='POSSIBILITE DE PLURIEL:';
    DETPOS:='POSSIBILITE DE DETERMINANT:';
    OUI:='OUI';
    NON:='NON';
    CLASSE:='CLASSE ';
    NRO:=1;
    CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASCREE);
    GOTOXY(3,15);WRITE('IMPRESSION EN COURS');
    WHILE NOT EOF(DICTIONNAIRE)  DO
      BEGIN
        SORTIE:=DICTIONNAIRE^;
        STR(NRO,STRNRO);
        UNITWRITE(6,TITRE[1],LENGTH(TITRE));
        UNITWRITE(6,BLANC,1);
        UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
        UNITWRITE(6,POINTS,1);
        UNITWRITE(6,SORTIE.MOT[1],LENGTH(SORTIE.MOT));
        FOR I:=1 TO 4 DO UNITWRITE(6,BLANC,1);
        UNITWRITE(6,GENRE[1],LENGTH(GENRE));
        IF SORTIE.GENRE THEN UNITWRITE(6,MASCULIN[1],LENGTH(MASCULIN))
                        ELSE UNITWRITE(6,FEMININ[1],LENGTH(FEMININ));
        FOR I:=1 TO 4 DO UNITWRITE(6,BLANC,1);
        UNITWRITE(6,PLURPOS[1],LENGTH(PLURPOS));
        IF SORTIE.POSPLUR THEN UNITWRITE(6,OUI[1],LENGTH(OUI))
                          ELSE UNITWRITE(6,NON[1],LENGTH(NON));
        UNITWRITE(6,CRRET,1);
        FOR I:=1 TO 3 DO UNITWRITE(6,BLANC,1);
        UNITWRITE(6,DETPOS[1],LENGTH(DETPOS));
        IF SORTIE.POSDET THEN UNITWRITE(6,OUI[1],LENGTH(OUI))
                         ELSE UNITWRITE(6,NON[1],LENGTH(NON));
        FOR I:=1 TO 4 DO UNITWRITE(6,BLANC,1);
        UNITWRITE(6,CLASSE[1],LENGTH(CLASSE));
        STR(SORTIE.CLASSE,STRNRO);
        UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
        IF SORTIE.PTRCLASSE > 0
          THEN
            BEGIN
              SUIVANT:=SORTIE.PTRCLASSE;
              STR(SUPPLEMENTAIRECLASSE[SUIVANT].NUMERO,STRNRO);
              UNITWRITE(6,BLANC,1);
```

```
              UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
              WHILE(SUPPLEMENTAIRECLASSE[SUIVANT].CLASPTR > 0) DO
                BEGIN
                  SUIVANT:=SUPPLEMENTAIRECLASSE[SUIVANT].CLASPTR;
                  STR(SUPPLEMENTAIRECLASSE[SUIVANT].NUMERO,STRNRO);
                  UNITWRITE(6,BLANC,1);
                  UNITWRITE(6,STRNRO[1],LENGTH(STRNRO))
                END
            END;
        UNITWRITE(6,CRRET,1);
        GET (DICTIONNAIRE);
        NRO:=NRO + 1
      END;(*WHILE NOT EOF*)
    GOTOXY(3,15);WRITE(VIDE);
    GOTOXY(3,15);WRITE('IMPRESSION TERMINEE');
  END;(*CORIMPRIM*)

BEGIN
   PAGE(OUTPUT);
   GOTOXY(3,6);WRITE('IMPRESSION DE LA LISTE ALPHABETIQUE');
   GOTOXY(14,8);WRITE('DES MOTS');
   GOTOXY(3,11);WRITE('VERIFIEZ SI L''IMPRIMANTE EST BRANCHEE');
   GOTOXY(3,13);WRITE('PUIS TAPEZ N''IMPORTE QUELLE TOUCHE.');
   READ(KEYBOARD,REP);
   (*$I-*)
   RESET (DICTIONNAIRE,'#5:DICTIO.DATA');
   IF IORESULT=0 (*$I+*)
    THEN
     BEGIN
      (*$I-*)
      UNITCLEAR(6);
      (*$I+*)
      IF IORESULT<>0
        THEN
          BEGIN
            GOTOXY(3,20);WRITE('IORESULT:',IORESULT);
            GOTOXY(3,15);
            WRITE('L''IMPRIMANTE N''EST PAS BRANCHEE!')
          END
        ELSE CORIMPRIM;
      CLOSE (DICTIONNAIRE,LOCK);
      GOTOXY(0,11);WRITE(VIDE)
     END
    ELSE
     BEGIN
      GOTOXY(3,11);
      FOR I:=1 TO 3 DO WRITELN(VIDE);
      GOTOXY(3,11);
      WRITE('PAS DE MOTS DANS LA LISTE.');
      GOTOXY(3,13);
      WRITE('CELLE-CI N''EXISTE PAS!');
     END;
  CONT
 END;(*LISTIMPRIM*)

BEGIN
  REP:='0';
  ACCEPTE:=['1','2','3','4','5'];
  WHILE REP<>'5'DO
    BEGIN
      PAGE(OUTPUT);
      GOTOXY(11,0);WRITE('MENU CONSULTATION');
      GOTOXY(10,1);FOR I:=1 TO 20 DO WRITE('-');
      GOTOXY(1,3);WRITE('1:LISTE ALPHABETIQUE DES MOTS A L''ECRAN');
      GOTOXY(1,6);WRITE('2:LISTE DES CLASSES DE MOTS EXISTANTES');
      GOTOXY(8,7);WRITE('A L''ECRAN');
      GOTOXY(1,10);WRITE('3:IMPRESSION DE LA LISTE DES MOTS');
      GOTOXY(1,13);WRITE('4:IMPRESSION DE LA LISTE DES CLASSES ');
```

```
      GOTOXY(13,14);WRITE('EXISTANTES');
      GOTOXY(1,17);WRITE('5:RETOUR AU MENU DE GESTION DE MOTS');
      GOTOXY(23,23);WRITE('VOTRE CHOIX:');
      READ (REP);WHILE NOT(REP IN ACCEPTE)DO
        BEGIN
          GOTOXY(4,20);WRITE('ATTENTION!');
          GOTOXY(4,21);WRITE('TAPEZ SEULEMENT UN NOMBRE COMPRIS');
          GOTOXY(6,22);WRITE('ENTRE 1 ET 5');
          GOTOXY(35,23);WRITE('      ');
          GOTOXY(35,23);READ(REP)
        END;
      CASE REP OF '1':LISTECRAN;
                  '2':CLASSEXIST;
                  '3':LISTIMPRIM;
                  '4':IMPRESCLAS
               END
    END
END;(*LISTING*)

BEGIN
END.
```

```
(*$S+,N+*)
UNIT U7;

INTERFACE

USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2,(*$U APPLE1:U4.CODE*)U4,
     (*$U APPLE1:U5.CODE*)U5;

PROCEDURE ENCORECLASSE(LIGNE:INTEGER;VAR ENCORE:CHAR);

PROCEDURE RECEPNROCLASSE(LISTNROCLASSE:SET150;COLONNE,LIGNE:INTEGER;
MODIF:BOOLEAN;VAR NROCLASSE,LONG:INTEGER);

PROCEDURE AJOUTDICT(ENTREE:TYPMOT;INDAP:INTEGER;VAR LISTMOT:TABCAR;
  VAR INDICE:TABIND;VAR LISTSUJET,LISTVERBE,LISTLIEU:LISTE;VAR
  SUPPLEMENTAIRECLASSE:LICLASSUP;VAR NBREMOT,NBRECAR,NBRESUJET,NBRELIEU,
  NBREVERBE,DERNCLASSUP:INTEGER);

IMPLEMENTATION

PROCEDURE ENCORECLASSE;
BEGIN
 GOTOXY(0,LIGNE);WRITE(VIDE);
 GOTOXY(0,23);WRITE(VIDE);
 GOTOXY(3,LIGNE);WRITE('ENCORE UNE CLASSE(O/N)?');READ(ENCORE);
 WHILE (ENCORE<>'O') AND (ENCORE<>'N')DO
   BEGIN
     GOTOXY(26,LIGNE);READ(ENCORE)
   END
END;(*ENCORECLASSE*)

PROCEDURE RECEPNROCLASSE;
VAR NUMEROCLASSE:CHAINE;
BEGIN
  GOTOXY(COLONNE,LIGNE);READLN(NUMEROCLASSE);
  IF (LENGTH(NUMEROCLASSE)=0)AND MODIF
   THEN NROCLASSE:=-1
   ELSE
    BEGIN
     CONVERSION(NUMEROCLASSE,NROCLASSE);
     WHILE NOT(NROCLASSE IN LISTNROCLASSE)AND(NROCLASSE<>201)AND
                                             (NROCLASSE<>301)AND(NROCLASSE<>401)DO
     BEGIN
      GOTOXY(COLONNE-1,LIGNE+1);WRITE('LA CLASSE ',NROCLASSE,' N''EXISTE PAS');
      GOTOXY(COLONNE,LIGNE);WRITE('       ');
      GOTOXY(COLONNE,LIGNE);READLN(NUMEROCLASSE);
      CONVERSION(NUMEROCLASSE,NROCLASSE)
     END
    END;
  LONG:=LENGTH(NUMEROCLASSE)+1
END;(*RECEPNROCLASSE*)

PROCEDURE AJOUTDICT;
VAR IND,INDCARAP,SUJETAP,LIEUAP,VERBAP,CLASSESUIVANTE,INDSUJET,INDVERBE,I,
     INDLIEU:INTEGER;
    ORTHO:BOOLEAN;

 PROCEDURE AJOUTLISTE(ENTREE:TYPMOT;MOTTAP,INDCARAP,NROCLASSE:INTEGER;VAR ENS:
                      LISTE;VAR NBRE:INTEGER);
  BEGIN
    NBRE:=NBRE+1;
    FOR IND:=NBRE DOWNTO (MOTTAP +1)DO
     BEGIN
      ENS[IND].INDICE:=ENS[IND-1].INDICE;
      ENS[IND].GENRE:=ENS[IND-1].GENRE;
      ENS[IND].POSPLUR:=ENS[IND-1].POSPLUR;
      ENS[IND].POSDET:=ENS[IND-1].POSDET;
      ENS[IND].CLASSE:=ENS[IND-1].CLASSE
```

```
      END;
     ENS[MOTTAP].INDICE:=INDCARAP;
     ENS[MOTTAP].GENRE:=ENTREE.GENRE;
     ENS[MOTTAP].POSPLUR:=ENTREE.POSPLUR;
     ENS[MOTTAP].POSDET:=ENTREE.POSDET;
     ENS[MOTTAP].CLASSE:=NROCLASSE
 END;(*AJOUTLISTE*)

 PROCEDURE AJOUTVERBE(ENTREE:TYPMOT;VERBAP,INDCARAP:INTEGER);
 BEGIN
  NBREVERBE:=NBREVERBE+1;
  FOR IND:=NBREVERBE DOWNTO (VERBAP +1)DO
   BEGIN
    LISTVERBE[IND].INDICE:=LISTVERBE[IND-1].INDICE;
    LISTVERBE[IND].CLASSE:=LISTVERBE[IND-1].CLASSE
   END;
  LISTVERBE[VERBAP].INDICE:=INDCARAP;
  LISTVERBE[VERBAP].CLASSE:=ENTREE.CLASSE;
END;(*AJOUTVERBE*)

BEGIN
  (*$R U1,U2*)
  NBREMOT:=NBREMOT+1;
  FOR IND:=NBREMOT DOWNTO (INDAP +1) DO
   BEGIN
    INDICE[IND].INDLIS:=INDICE[IND-1].INDLIS+LENGTH(ENTREE.MOT)+1;
    INDICE[IND].PTRTYP:=INDICE[IND-1].PTRTYP
   END;
  NBRECAR:=NBRECAR+LENGTH(ENTREE.MOT)+1;
  FOR IND:=NBRECAR DOWNTO (INDICE[INDAP+1].INDLIS-1) DO
      LISTMOT[IND]:=LISTMOT[IND-LENGTH(ENTREE.MOT)-1];
  INDCARAP:=INDICE[INDAP].INDLIS;
  RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCARAP,ORTHO,INDSUJET,SUJETAP);
  FOR IND:=SUJETAP TO NBRESUJET DO LISTSUJET[IND].INDICE:=LISTSUJET[IND].INDICE
                                   +LENGTH(ENTREE.MOT)+1;
  RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCARAP,ORTHO,INDLIEU,LIEUAP);
  FOR IND:=LIEUAP TO NBRELIEU DO LISTLIEU[IND].INDICE:=LISTLIEU[IND].INDICE
                                   +LENGTH(ENTREE.MOT)+1;
  RECHSUVERLIEU(LISTVERBE,NBREVERBE,INDCARAP,ORTHO,INDVERBE,VERBAP);
  FOR IND:=VERBAP TO NBREVERBE DO LISTVERBE[IND].INDICE:=LISTVERBE[IND].INDICE
                                                   +LENGTH(ENTREE.MOT)+1;
  MOVELEFT(ENTREE.MOT[1],LISTMOT[INDCARAP],LENGTH(ENTREE.MOT));
  IF ENTREE.CLASSE>=200
   THEN
   BEGIN
     AJOUTVERBE(ENTREE,VERBAP,INDCARAP);
     INDICE[INDAP].PTRTYP:='V'
   END
   ELSE
    BEGIN
     IF ENTREE.CLASSE<=100
      THEN
       BEGIN
         AJOUTLISTE(ENTREE,SUJETAP,INDCARAP,ENTREE.CLASSE,LISTSUJET,NBRESUJET);
         SUJETAP:=SUJETAP+1;
         INDICE[INDAP].PTRTYP:='S'
       END
      ELSE
       BEGIN
        AJOUTLISTE(ENTREE,LIEUAP,INDCARAP,ENTREE.CLASSE,LISTLIEU,NBRELIEU);
        LIEUAP:=LIEUAP+1;
        INDICE[INDAP].PTRTYP:='L'
       END;
     CLASSESUIVANTE:=ENTREE.PTRCLASSE;
     WHILE CLASSESUIVANTE>0 DO
      BEGIN
       IF SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO<=100
        THEN
```

```
            BEGIN
          AJOUTLISTE(ENTREE,SUJETAP,INDCARAP,
            SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO,LISTSUJET,NBRESUJET);
           IF INDICE[INDAP].PTRTYP='L' THEN INDICE[INDAP].PTRTYP:='D'
            END
         ELSE
            BEGIN
           AJOUTLISTE(ENTREE,LIEUAP,INDCARAP,
              SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].NUMERO,LISTLIEU,NBRELIEU);
           IF INDICE[INDAP].PTRTYP='S' THEN INDICE[INDAP].PTRTYP:='D'
            END;
        CLASSESUIVANTE:=SUPPLEMENTAIRECLASSE[CLASSESUIVANTE].CLASPTR;
       END
    END
END;(*AJOUTDICT*)

BEGIN
END.
```

```
(*$S+,N+*)
UNIT U7B;

INTERFACE
USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2,(*$U APPLE1:U4.CODE*)U4,
     (*$U APPLE1:U5.CODE*)U5,(*$U APPLE2:U7.CODE*)U7;

PROCEDURE AJOUT;

IMPLEMENTATION

PROCEDURE AJOUT;

VAR REP,GENRE,PLURIEL,DETERMINANT,ENCORE:CHAR;
    ENTREE:TYPMOT;
    MODIF,PASCLAS,PASSUP,PASDICT,DEUXIEME,ORTHO,FINI:BOOLEAN;
    NROCLASSE,LIGNE,COLONNE,INDAP,INDMOT,I,DERNCLASSUP,NBRECLASSE,
     NBCLASUJET,NBCLALIEU,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,LONG:
       INTEGER;
    SUPPLEMENTAIRECLASSE:LICLASSUP;
    CLASEXIST:EXICLASSE;
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;
    LISTMOT:TABCAR;
    LISTSUJET,LISTVERBE,LISTLIEU:LISTE;
    INDICE:TABIND;

PROCEDURE CORPAJOUT;
BEGIN
  GOTOXY(0,23);WRITE('TERMINER VOTRE REPONSE PAR "RETURN"!');
  GOTOXY(0,2);WRITE('MOT A AJOUTER:');READLN(ENTREE.MOT);
  RECHERCHE(LISTMOT,INDICE,NBREMOT,ENTREE.MOT,ORTHO,INDMOT,INDAP);
  IF ORTHO
    THEN
      BEGIN
       GOTOXY(2,3);
       WRITE(ENTREE.MOT,' EXISTE DEJA AU DICTIONNAIRE')
      END
    ELSE
      BEGIN
        GOTOXY(0,23);WRITE(VIDE);
        GOTOXY(27,2);WRITE('GENRE(M/F):');
        READ(GENRE);
        WHILE(GENRE<>'M')AND(GENRE<>'F')DO
          BEGIN
            GOTOXY(38,2);READ(GENRE)
          END;
        IF GENRE='M'THEN ENTREE.GENRE:=TRUE
                    ELSE ENTREE.GENRE:=FALSE;
        GOTOXY(2,3);WRITE('POSSIBILITE DE PLURIEL(O/N):');
        READ(PLURIEL);
        WHILE(PLURIEL<>'O')AND(PLURIEL<>'N')DO
          BEGIN
            GOTOXY(30,3);READ(PLURIEL)
          END;
        IF PLURIEL='O'THEN ENTREE.POSPLUR:=TRUE
                      ELSE ENTREE.POSPLUR:=FALSE;
        GOTOXY(2,4);WRITE('POSSIBILITE DE DETERMINANT(O/N):');
        READ(DETERMINANT);
        WHILE(DETERMINANT<>'O')AND(DETERMINANT<>'N')DO
          BEGIN
            GOTOXY(34,4);READ(DETERMINANT)
          END;
        IF DETERMINANT='O'THEN ENTREE.POSDET:=TRUE
                          ELSE ENTREE.POSDET:=FALSE;
        GOTOXY(2,5);WRITE('NUMERO DE CLASSE:');
        GOTOXY(0,23);WRITE('TERMINER VOTRE REPONSE PAR "RETURN"!');
```

```
        MODIF:=FALSE;
        RECEPNROCLASSE(LISTNROCLASSE,19,5,MODIF,NROCLASSE,LONG);
        ENTREE.CLASSE:=NROCLASSE;
        IF NROCLASSE < 200
         THEN
          BEGIN
           COLONNE:=19;
           DEUXIEME:=TRUE;
           ENCORECLASSE(6,ENCORE);
           IF ENCORE = 'N'
            THEN ENTREE.PTRCLASSE:=0
            ELSE WHILE ENCORE = 'O' DO
             BEGIN
               COLONNE:=COLONNE+LONG;
               GOTOXY(0,23);WRITE('TERMINER VOTRE REPONSE PAR "RETURN"');
               GOTOXY(COLONNE,5);WRITE('?');
               RECEPNROCLASSE(LISTNROCLASSE,COLONNE,5,MODIF,NROCLASSE,LONG);
               DERNCLASSUP:=DERNCLASSUP+1;
               IF DEUXIEME THEN ENTREE.PTRCLASSE:=DERNCLASSUP
                           ELSE SUPPLEMENTAIRECLASSE[DERNCLASSUP-1].CLASPTR
                                :=DERNCLASSUP;
               SUPPLEMENTAIRECLASSE[DERNCLASSUP].NUMERO:=NROCLASSE;
               DEUXIEME:=FALSE;
               ENCORECLASSE(6,ENCORE)
             END;
           IF NOT DEUXIEME THEN SUPPLEMENTAIRECLASSE[DERNCLASSUP].CLASPTR:=0
          END;
        AJOUTDICT(ENTREE,INDAP,LISTMOT,INDICE,LISTSUJET,LISTVERBE,LISTLIEU
                  ,SUPPLEMENTAIRECLASSE,NBREMOT,NBRECAR,NBRESUJET,NBRELIEU
                  ,NBREVERBE,DERNCLASSUP)
      END;
    CONTARRET(2,7,FINI)
END;(*CORPAJOUT*)

BEGIN
    (*$R U1,U2*)
    PAGE(OUTPUT);
    GOTOXY(11,0);WRITE('AJOUT DE MOT(S)');
    GOTOXY(10,1);FOR I:=1 TO 16 DO WRITE('-');
    LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
       LINOCLASU,LINOCLALI,PASCLAS);
    LIGNE:=10;
    COLONNE:=0;
    IF NOT PASCLAS
     THEN
      BEGIN
       ECRCLASSEX(CLASEXIST,NBRECLASSE,21,COLONNE,LIGNE);
       CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASSUP);
       CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE,DERNCLASSUP,LISTMOT,LISTSUJET,
          LISTLIEU,LISTVERBE,INDICE,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,
          PASDICT);
       FINI:=FALSE;
       WHILE NOT FINI DO CORPAJOUT;
       SAUVETAGE(LISTMOT,INDICE,LISTSUJET,LISTLIEU,LISTVERBE,NBRECAR,NBREMOT,
                 NBRESUJET,NBREVERBE,NBRELIEU,SUPPLEMENTAIRECLASSE,DERNCLASSUP);
       SAUVECLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP)
      END
     ELSE
      BEGIN
       GOTOXY(0,10);
       WRITE('PAS DE CLASSES EXISTANTES!');
       CONT
      END
END;(*AJOUT*)

BEGIN
END.
```

```
(*$S+,N+*)

UNIT U8;

INTERFACE
USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2,(*$U APPLE1:U4.CODE*)U4,
     (*$U APPLE1:U5.CODE*)U5;
PROCEDURE SUPPRESDICT(ENTREE:CHAINE;INDMOT:INTEGER;VAR LISTMOT:TABCAR;
                      VAR INDICE:TABIND;VAR LISTSUJET,LISTVERBE,LISTLIEU:LISTE;
                      VAR NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU:INTEGER);

IMPLEMENTATION

PROCEDURE SUPPRESDICT;
VAR IND,INDCAR,INDVERBE,VERBAP,INDSUJET,SUJETAP,INDLIEU,LIEUAP:INTEGER;
    ORTHO:BOOLEAN;

 PROCEDURE SUPPRELISTE(INDCAR,MARK:INTEGER;VAR ENS:LISTE;VAR DIMENS:INTEGER);
 VAR INDMOT,NBRESUPPRES,DEPART:INTEGER;
 BEGIN
   NBRESUPPRES:=0;
   INDMOT:=MARK;
   WHILE INDMOT>0 DO
    IF ENS[INDMOT].INDICE=INDCAR
     THEN
      BEGIN
       NBRESUPPRES:=NBRESUPPRES+1;
       DEPART:=INDMOT;
       INDMOT:=INDMOT-1
      END
     ELSE INDMOT:=0;
   INDMOT:=MARK+1;
   WHILE INDMOT<=DIMENS DO
    IF ENS[INDMOT].INDICE=INDCAR
     THEN
      BEGIN
       NBRESUPPRES:=NBRESUPPRES+1;
       INDMOT:=INDMOT+1
      END
     ELSE INDMOT:=DIMENS+1;
   DIMENS:=DIMENS-NBRESUPPRES;
   FOR IND:=DEPART TO DIMENS DO
     BEGIN
      ENS[IND].INDICE:=ENS[IND+NBRESUPPRES].INDICE-
                                          (LENGTH(ENTREE)*NBRESUPPRES)-NBRESUPPRES;
      ENS[IND].GENRE:=ENS[IND+NBRESUPPRES].GENRE;
      ENS[IND].POSPLUR:=ENS[IND+NBRESUPPRES].POSPLUR;
      ENS[IND].POSDET:=ENS[IND+NBRESUPPRES].POSDET;
      ENS[IND].CLASSE:=ENS[IND+NBRESUPPRES].CLASSE
     END
 END;(*SUPPRELISTE*)

 BEGIN
  (*$R U1,U2*)
  INDCAR:=INDICE[INDMOT].INDLIS;
  NBREMOT:=NBREMOT-1;
  FOR IND:=INDMOT TO NBREMOT DO
    BEGIN
     INDICE[IND].INDLIS:=INDICE[IND+1].INDLIS-LENGTH(ENTREE)-1;
     INDICE[IND].PTRTYP:=INDICE[IND+1].PTRTYP
    END;
  NBRECAR:=NBRECAR-LENGTH(ENTREE)-1;
  FOR IND:=INDICE[INDMOT].INDLIS TO NBRECAR DO
     LISTMOT[IND]:=LISTMOT[IND+LENGTH(ENTREE)+1];
  RECHSUVERLIEU(LISTVERBE,NBREVERBE,INDCAR,ORTHO,INDVERBE,VERBAP);
  IF ORTHO
    THEN
      BEGIN
```

```
        NBREVERBE:=NBREVERBE-1;
        FOR IND:=INDVERBE TO NBREVERBE DO
         BEGIN
          LISTVERBE[IND].INDICE:=LISTVERBE[IND+1].INDICE-LENGTH(ENTREE)-1;
          LISTVERBE[IND].CLASSE:=LISTVERBE[IND+1].CLASSE
         END
      END
    ELSE FOR IND:=VERBAP TO NBREVERBE DO
          LISTVERBE[IND].INDICE:=LISTVERBE[IND].INDICE-LENGTH(ENTREE)-1;
  RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
  IF NOT ORTHO
    THEN FOR IND:=SUJETAP TO NBRESUJET DO
      LISTSUJET[IND].INDICE:=LISTSUJET[IND].INDICE-LENGTH(ENTREE)-1
    ELSE SUPPRELISTE(INDCAR,INDSUJET,LISTSUJET,NBRESUJET);
  RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
  IF NOT ORTHO
    THEN FOR IND:=LIEUAP TO NBRELIEU DO
      LISTLIEU[IND].INDICE:=LISTLIEU[IND].INDICE-LENGTH(ENTREE)-1
    ELSE SUPPRELISTE(INDCAR,INDLIEU,LISTLIEU,NBRELIEU)
 END
END;(*SUPPRESDICT*)

BEGIN
END.
```

```
(*$S+,N+*)
UNIT U8B;

INTERFACE
 USES (*$U APPLE1:U1.CODE*)U1,(*$U APPLE1:U2.CODE*)U2,(*$U APPLE1:U4.CODE*)U4,
       (*$U APPLE1:U5.CODE*)U5,(*$U APPLE2:U8.CODE*)U8;

 PROCEDURE SUPPRESSION;

IMPLEMENTATION

PROCEDURE SUPPRESSION;
VAR I,LIGNE,COLONNE,INDMOT,INDAP,INDCAR,INDVERBE,VERBAP,INDSUJET,SUJETAP,
     INDLIEU,LIEUAP,DERNCLASSUP,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU:
       INTEGER;
    PASDICT,PASSUP,FINI,ORTHO:BOOLEAN;
    REP:CHAR;
    ENTREE:CHAINE;
    SUPPLEMENTAIRECLASSE:LICLASSUP;
    LISTMOT:TABCAR;
    LISTSUJET,LISTLIEU,LISTVERBE:LISTE;
    INDICE:TABIND;

PROCEDURE CONSUPPRES(DEPOUI,DEPNON:INTEGER);
VAR REP:CHAR;
BEGIN
  GOTOXY(2,LIGNE);WRITE('ETES-VOUS SUR QU''IL FAUT LE SUPPRIMER?');
  GOTOXY(2,LIGNE+1);WRITE('REPONDEZ PAR (O)UI OU PAR (N)ON:');
  READ(KEYBOARD,REP);
  WHILE(REP<>'O')AND(REP<>'N')DO
   BEGIN
    GOTOXY(34,LIGNE+1);
    READ(KEYBOARD,REP)
   END;
  IF REP='O'
   THEN
    BEGIN
     GOTOXY(2,LIGNE+2);
     WRITE(ENTREE,' N''EST PLUS AU DICTIONNAIRE');
     SUPPRESDICT(ENTREE,INDMOT,LISTMOT,INDICE,LISTSUJET,LISTVERBE,LISTLIEU,
                 NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU);
     LIGNE:=LIGNE+DEPOUI
    END
   ELSE LIGNE:=LIGNE+DEPNON
END;(*CONSUPPRES*)

BEGIN
 (*$R U1,U2*)
 PAGE(OUTPUT);
 GOTOXY(9,0);WRITE('SUPPRESSION DE MOT(S)');
 GOTOXY(8,1);FOR I:=1 TO 23 DO WRITE('-');
 CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASSUP);
 CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE,DERNCLASSUP,LISTMOT,LISTSUJET,
       LISTLIEU,LISTVERBE,INDICE,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,
       PASDICT);
 IF NOT PASDICT
  THEN
   BEGIN
    GOTOXY(0,23);WRITE('TERMINEZ VOTRE REPONSE PAR"RETURN"');
    FINI:=FALSE;
    LIGNE:=2;
    WHILE NOT FINI DO
     BEGIN
      GOTOXY(0,LIGNE);
      WRITE('MOT A SUPPRIMER:');READLN(ENTREE);
      RECHERCHE(LISTMOT,INDICE,NBREMOT,ENTREE,ORTHO,INDMOT,INDAP);
      IF NOT ORTHO
       THEN
```

```
        BEGIN
         GOTOXY(2,LIGNE+1);
         WRITE(ENTREE,' N''EST PAS AU DICTIONNAIRE.');
         GOTOXY(0,LIGNE);WRITE(VIDE);
         LIGNE:=LIGNE+3
        END
       ELSE
        BEGIN
         INDCAR:=INDICE[INDMOT].INDLIS;
         CASE INDICE[INDMOT].PTRTYP OF
          'V':BEGIN
               RECHSUVERLIEU(LISTVERBE,NBREVERBE,INDCAR,ORTHO,INDVERBE,VERBAP);
               VERBECARAC(LISTVERBE,INDVERBE,LIGNE);
               CONSUPPRES(6,7)
              END;
          'S':BEGIN
               RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
               MOTCARAC(LISTSUJET,NBRESUJET,INDSUJET,LIGNE);
               CONSUPPRES(9,8)
              END;
          'L':BEGIN
               RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
               MOTCARAC(LISTLIEU,NBRELIEU,INDLIEU,LIGNE);
               CONSUPPRES(9,8)
              END;
          'D':BEGIN
               RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
               MOTCARAC(LISTSUJET,NBRESUJET,INDSUJET,LIGNE);
               RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
               SULIEUCARAC(LISTLIEU,NBRELIEU,INDLIEU);
               CONSUPPRES(9,8)
              END
            END
         END;
        IF LIGNE>=15 THEN LIGNE:=2;
        CONTARRET(LIGNE,LIGNE+8,FINI)
       END;(*WHILE NOT FINI*)
      SAUVETAGE(LISTMOT,INDICE,LISTSUJET,LISTLIEU,LISTVERBE,NBRECAR,NBREMOT,
                NBRESUJET,NBREVERBE,NBRELIEU,SUPPLEMENTAIRECLASSE,
                DERNCLASSUP);
      SAUVECLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP)
     END
    ELSE
     BEGIN
      GOTOXY(0,9);
      WRITE('IL N"Y A PAS DE MOT DANS LE DICTIONNAIRE');
      GOTOXY(0,11);
      WRITE('ON NE PEUT DONC EN SUPPRIMER.');
      CONT
     END
END;(*SUPPRESSION*)

BEGIN
END.
```

```
(*$S+,N+*)

UNIT U9;

INTERFACE
 USES (*$U *:U1.CODE*)U1,(*$U *:U2.CODE*)U2,(*$U *:U4.CODE*)U4,
    (*$U *:U5.CODE*)U5,(*$U APPLE2:U7.CODE*)U7,(*$U APPLE2:U8.CODE*)U8;

PROCEDURE MODIFICATION;

IMPLEMENTATION

PROCEDURE MODIFICATION;
VAR I,LIGNE,INDMOT,INDAP,INDVERBE,VERBAP,INDSUJET,SUJETAP,COLONNE,
     LIEUAP,INDLIEU,INDCAR,LIMITE,DERNCLASSUP,NBREMOT,NBRECAR,NBRESUJET,
     NBRECLASSE,NBCLASUJET,NBCLALIEU,NBREVERBE,NBRELIEU:INTEGER;
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;
    CLASEXIST:EXICLASSE;
    NOUVEAU:TYPMOT;
    PASCLAS,LONGNUL,PASCREE,FINI,ORTHO:BOOLEAN;
    ENTREE:CHAINE;
    REP:CHAR;
    SUPPLEMENTAIRECLASSE:LICLASSUP;
    LISTMOT:TABCAR;
    LISTSUJET,LISTLIEU,LISTVERBE:LISTE;
    INDICE:TABIND;

PROCEDURE MODIVERBE(INDVERBE,LIGNE:INTEGER);
VAR MODIF:CHAINE;
BEGIN
  GOTOXY(0,22);WRITELN('TAPEZ"RETURN"SI VOUS NE MODIFIEZ PAS');
  WRITE('SINON ECRIVEZ D''ABORD LA MODIFICATION');
  LIGNE:=2;
  GOTOXY(15,LIGNE);READLN(MODIF);
  IF LENGTH(MODIF)>0 THEN NOUVEAU.MOT:=MODIF
                     ELSE NOUVEAU.MOT:=ENTREE;
  GOTOXY(9,LIGNE+1);READLN(MODIF);
  IF LENGTH(MODIF)=0
   THEN NOUVEAU.CLASSE:=LISTVERBE[INDVERBE].CLASSE
   ELSE
    BEGIN
     WHILE(MODIF<>'201')AND(MODIF<>'301')AND(MODIF<>'401')DO
      BEGIN
       GOTOXY(9,LIGNE+1);WRITE('        ');
       GOTOXY(9,LIGNE+1);READLN(MODIF)
      END;
     IF MODIF ='201'THEN NOUVEAU.CLASSE:=201
                    ELSE IF MODIF ='301' THEN NOUVEAU.CLASSE:=301
                                         ELSE NOUVEAU.CLASSE:=401
    END
 END;(*MODIVERBE*)

PROCEDURE MODIMOT(ENS:LISTE;DIM:INTEGER;VAR INDMOT,LIGNE:INTEGER);
VAR NROMODIF,LONG,COLONNE:INTEGER;
    MODIF:CHAINE;
    ENCORE:BOOLEAN;

 PROCEDURE MODIPHONEME;
 BEGIN
  GOTOXY(15,LIGNE);READLN(MODIF);
  IF LENGTH(MODIF)>0 THEN NOUVEAU.MOT:=MODIF
                     ELSE NOUVEAU.MOT:=ENTREE;
 END;(*MODIPHONEME*)

 PROCEDURE MODIGENRE;
 BEGIN
```

```
 GOTOXY(13,LIGNE+1);READLN(MODIF);
 IF LENGTH(MODIF)=0
   THEN NOUVEAU.GENRE:=ENS[INDMOT].GENRE
   ELSE
    BEGIN
     WHILE(MODIF<>'M')AND(MODIF<>'F')DO
      BEGIN
       GOTOXY(13,LIGNE+1);WRITE('         ');
       GOTOXY(13,LIGNE+1);READLN(MODIF)
      END;
     GOTOXY(14,LIGNE+1);
     IF MODIF='M'
      THEN
       BEGIN
        WRITE('ASCULIN');NOUVEAU.GENRE:=TRUE
       END
      ELSE
       BEGIN
        WRITE('EMININ');NOUVEAU.GENRE:=FALSE
       END
    END
END;(*MODIGENRE*)

PROCEDURE MODIPOSPLUR;
BEGIN
  GOTOXY(30,LIGNE+2);READLN(MODIF);
  IF LENGTH(MODIF)=0
   THEN NOUVEAU.POSPLUR:=ENS[INDMOT].POSPLUR
   ELSE
    BEGIN
     WHILE(MODIF<>'O')AND(MODIF<>'N')DO
      BEGIN
       GOTOXY(30,LIGNE+2);WRITE('         ');
       GOTOXY(30,LIGNE+2);READLN(MODIF)
      END;
     GOTOXY(31,LIGNE+2);
     IF MODIF='O'
      THEN
       BEGIN
        WRITE('UI');NOUVEAU.POSPLUR:=TRUE
       END
      ELSE
       BEGIN
        WRITE('ON');NOUVEAU.POSPLUR:=FALSE
       END
    END
END;(*MODIPOSPLUR*)

PROCEDURE MODIPOSDET;
BEGIN
  GOTOXY(34,LIGNE+3);READLN(MODIF);
  IF LENGTH(MODIF)=0
   THEN NOUVEAU.POSDET:=ENS[INDMOT].POSDET
   ELSE
    BEGIN
     WHILE(MODIF<>'O')AND(MODIF<>'N')DO
      BEGIN
       GOTOXY(34,LIGNE+3);WRITE('         ');
       GOTOXY(34,LIGNE+3);READLN(MODIF)
      END;
     GOTOXY(35,LIGNE+3);
     IF MODIF='O'
      THEN
       BEGIN
        WRITE('UI');NOUVEAU.POSDET:=TRUE
       END
      ELSE
       BEGIN
```

```
          WRITE('ON');NOUVEAU.POSDET:=FALSE
         END
      END
 END;(*MODIPOSDET*)

 PROCEDURE MODICLASSE(VAR NROMODIF:INTEGER);
 BEGIN
   RECEPNROCLASSE(LISTNROCLASSE,9,LIGNE+4,LONGNUL,NROMODIF,LONG);
   IF NROMODIF<1
    THEN NOUVEAU.CLASSE:=ENS[INDMOT].CLASSE
    ELSE NOUVEAU.CLASSE:=NROMODIF
 END;(*MODICLASSE*)

 PROCEDURE MODISUPPLECLASSE(NROMODIF:INTEGER);
 VAR DEUXIEME:BOOLEAN;
     ENCORE:CHAR;
 BEGIN
   LIGNE:=LIGNE+4;
   COLONNE:=10+LONG;
   GOTOXY(COLONNE,LIGNE);WRITE(VIDE);
   DEUXIEME:=TRUE;
   ENCORECLASSE(LIGNE+1,ENCORE);
   IF ENCORE ='N'
    THEN NOUVEAU.PTRCLASSE:=0
    ELSE WHILE ENCORE ='O' DO
          BEGIN
           GOTOXY(0,23);WRITE('TERMINEZ VOTRE REPONSE PAR "RETURN"');
           GOTOXY(COLONNE,LIGNE);WRITE(VIDE);
           GOTOXY(COLONNE,LIGNE);WRITE('?');
           DERNCLASSUP:=DERNCLASSUP+1;
           RECEPNROCLASSE(LISTNROCLASSE,COLONNE,LIGNE,LONGNUL,
                                SUPPLEMENTAIRECLASSE[DERNCLASSUP].NUMERO,LONG);
           IF DEUXIEME THEN NOUVEAU.PTRCLASSE:=DERNCLASSUP
                       ELSE SUPPLEMENTAIRECLASSE[DERNCLASSUP-1].CLASPTR:=
                                 DERNCLASSUP;
           DEUXIEME:=FALSE;
           ENCORECLASSE(LIGNE+1,ENCORE);
           COLONNE:=COLONNE+LONG;
           IF NOT DEUXIEME THEN SUPPLEMENTAIRECLASSE[DERNCLASSUP].CLASPTR:=0
          END
 END;(*MODISUPPLECLASSE*)

 BEGIN
  GOTOXY(0,22);WRITELN('TAPEZ"RETURN"SI VOUS NE MODIFIEZ PAS');
  WRITE('SINON,ECRIVEZ D''ABORD LA MODIFICATION.');
  LIGNE:=2;
  LONGNUL:=TRUE;
  MODIPHONEME;
  MODIGENRE;
  MODIPOSPLUR;
  MODIPOSDET;
  MODICLASSE(NROMODIF);
  MODISUPPLECLASSE(NROMODIF)
 END;(*MODIMOT*)

PROCEDURE ORTHOVRAI;
BEGIN
  INDCAR:=INDICE[INDMOT].INDLIS;
  CASE INDICE[INDMOT].PTRTYP OF
  'V':BEGIN
       RECHSUVERLIEU(LISTVERBE,NBREVERBE,INDCAR,ORTHO,INDVERBE,VERBAP);
       VERBECARAC(LISTVERBE,INDVERBE,LIGNE);
       MODIVERBE(INDVERBE,LIGNE)
      END;
  'S':BEGIN
       RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
       MOTCARAC(LISTSUJET,NBRESUJET,INDSUJET,LIGNE);
       INDSUJET:=INDSUJET-1;
```

```
        MODIMOT(LISTSUJET,NBRESUJET,INDSUJET,LIGNE)
          END;
  'L':BEGIN
        RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
        MOTCARAC(LISTLIEU,NBRELIEU,INDLIEU,LIGNE);
        INDLIEU:=INDLIEU-1;
        MODIMOT(LISTLIEU,NBRELIEU,INDLIEU,LIGNE)
      END;
  'D':BEGIN
        RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
        MOTCARAC(LISTSUJET,NBRESUJET,INDSUJET,LIGNE);
        RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
        SULIEUCARAC(LISTLIEU,NBRELIEU,INDLIEU);
        INDSUJET:=INDSUJET-1;
        MODIMOT(LISTSUJET,NBRESUJET,INDSUJET,LIGNE)
      END
  END;(*CASE*)
  SUPPRESDICT(ENTREE,INDMOT,LISTMOT,INDICE,LISTSUJET,LISTVERBE,LISTLIEU,
      NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU);
  RECHERCHE(LISTMOT,INDICE,NBREMOT,NOUVEAU.MOT,ORTHO,INDMOT,INDAP);
  AJOUTDICT(NOUVEAU,INDAP,LISTMOT,INDICE,LISTSUJET,LISTVERBE,LISTLIEU,
SUPPLEMENTAIRECLASSE,NBREMOT,NBRECAR,NBRESUJET,NBRELIEU,NBREVERBE,DERNCLASSUP);
 END;(*ORTHO GENERAL*)

BEGIN
 (*$R U1,U2*)
 PAGE(OUTPUT);
 GOTOXY(9,0);WRITE('MODIFICATION DE MOT(S)');
 GOTOXY(8,1);FOR I:=1 TO 24 DO WRITE('-');
 CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASCREE);
 CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE,DERNCLASSUP,LISTMOT,LISTSUJET,
    LISTLIEU,LISTVERBE,INDICE,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,
    PASCREE);
 IF NOT PASCREE
  THEN
   BEGIN
    LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
       LINOCLASU,LINOCLALI,PASCLAS);
    LIGNE:=8;
    IF NOT PASCLAS THEN ECRCLASSEX(CLASEXIST,NBRECLASSE,21,COLONNE,LIGNE)
                   ELSE
                    BEGIN
                     GOTOXY(0,LIGNE);
                     WRITE('PAS DE CLASSES EXISTANTES')
                    END;
    GOTOXY(0,23);WRITE('TERMINEZ VOTRE REPONSE PAR"RETURN"');
    FINI:=FALSE;
    WHILE NOT FINI DO
     BEGIN
      LIGNE:=2;
      GOTOXY(0,LIGNE);
      WRITE('MOT A MODIFIER:');READLN(ENTREE);
      RECHERCHE(LISTMOT,INDICE,NBREMOT,ENTREE,ORTHO,INDMOT,INDAP);
      IF NOT ORTHO
       THEN
        BEGIN
         GOTOXY(2,LIGNE+1);
         WRITE(ENTREE,' N''EST PAS AU DICTIONNAIRE.');
         GOTOXY(0,LIGNE);WRITE(VIDE)
        END
       ELSE ORTHOVRAI;
      GOTOXY(0,22);WRITE(VIDE);
      GOTOXY(0,23);WRITE(VIDE);
      LIGNE:=2;
      CONTARRET(LIGNE,7,FINI)
     END;(*WHILE NOT FINI*)
    SAUVETAGE(LISTMOT,INDICE,LISTSUJET,LISTLIEU,LISTVERBE,NBRECAR,NBREMOT,
                NBRESUJET,NBREVERBE,NBRELIEU,SUPPLEMENTAIRECLASSE,
```

```
                    DERNCLASSUP);
      SAUVECLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP)
     END
    ELSE
     BEGIN
      GOTOXY(0,9);
      WRITE('PAS DE MOTS DANS LE DICTIONNAIRE!');
      GOTOXY(0,11);
      WRITE('ON NE PEUT DONC EN MODIFIER.');
      CONT
     END
END;(*MODIFICATION*)

BEGIN
END.
```

```
(*$S+,N+*)

UNIT U10;

INTERFACE

 USES (*$U *:U1.CODE*)U1,(*$U *:U2.CODE*)U2,(*$U *:U4.CODE*)U4;

 PROCEDURE CONSULTATION;

 PROCEDURE CRECLASSE;

IMPLEMENTATION

PROCEDURE CONSULTATION;

VAR ENTREE:CHAINE;
    REP:CHAR;
    PASCREE,FINI,ORTHO:BOOLEAN;
    LIGNE,I,INDMOT,INDAP,INDCAR,INDVERBE,VERBAP,INDSUJET,SUJETAP,INDLIEU,
      LIEUAP,COLONNE,DERNCLASSUP,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU:
        INTEGER;
    SUPPLEMENTAIRECLASSE:LICLASSUP;
    LISTMOT:TABCAR;
    LISTSUJET,LISTLIEU,LISTVERBE:LISTE;
    INDICE:TABIND;

BEGIN
  (*$R U1,U2*)
  PAGE(OUTPUT);
  GOTOXY(9,0);
  WRITE('CONSULTATION DE MOT(S)');
  GOTOXY(8,1);
  FOR I:=1 TO 24 DO WRITE('-');
  CHARGCLASSUP(SUPPLEMENTAIRECLASSE,DERNCLASSUP,PASCREE);
  CHARGEMENTDICTIONNAIRE(SUPPLEMENTAIRECLASSE,DERNCLASSUP,LISTMOT,LISTSUJET,
       LISTLIEU,LISTVERBE,INDICE,NBREMOT,NBRECAR,NBRESUJET,NBREVERBE,NBRELIEU,
       PASCREE);
  IF NOT PASCREE
   THEN
    BEGIN
     GOTOXY(2,23);
     WRITE('TAPEZ"RETURN"POUR TERMINER UN MOT');
     FINI:=FALSE;
     LIGNE:=2;
     WHILE NOT FINI DO
      BEGIN
       GOTOXY(0,LIGNE);
       FOR I:=1 TO 3 DO WRITELN(VIDE);
       GOTOXY(0,LIGNE);
       WRITE('MOT A CONSULTER? ');
       READLN(ENTREE);
       RECHERCHE(LISTMOT,INDICE,NBREMOT,ENTREE,ORTHO,INDMOT,INDAP);
       IF NOT ORTHO
        THEN
         BEGIN
          GOTOXY(2,LIGNE+1);
          WRITE(ENTREE,' N"EST PAS DANS LE DICTIONNAIRE');
          LIGNE:=LIGNE+3
         END
        ELSE
         BEGIN
          INDCAR:=INDICE[INDMOT].INDLIS;
          CASE INDICE[INDMOT].PTRTYP OF
          'V':BEGIN
                RECHSUVERLIEU(LISTVERBE,NBREVERBE,INDCAR,ORTHO,INDVERBE,VERBAP);
                VERBECARAC(LISTVERBE,INDVERBE,LIGNE)
              END;
```

```
            'S':BEGIN
                RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
                MOTCARAC(LISTSUJET,NBRESUJET,INDSUJET,LIGNE)
                END;
            'L':BEGIN
                 RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
                 MOTCARAC(LISTLIEU,NBRELIEU,INDLIEU,LIGNE)
                END;
            'D':BEGIN
                RECHSUVERLIEU(LISTSUJET,NBRESUJET,INDCAR,ORTHO,INDSUJET,SUJETAP);
                MOTCARAC(LISTSUJET,NBRESUJET,INDSUJET,LIGNE);
                RECHSUVERLIEU(LISTLIEU,NBRELIEU,INDCAR,ORTHO,INDLIEU,LIEUAP);
                SULIEUCARAC(LISTLIEU,NBRELIEU,INDLIEU)
               END
             END
          END;
        IF LIGNE>=15 THEN LIGNE:=2;
        CONTARRET(LIGNE,LIGNE+6,FINI)
      END(*WHILE NOT FINI*)
    END
   ELSE
    BEGIN
     GOTOXY(0,9);
     WRITE('PAS DE MOTS A CONSULTER DANS LE ');
     GOTOXY(3,11);
     WRITE('DICTIONNAIRE!');
     CONT
    END
END;(*CONSULTATION*)

PROCEDURE CRECLASSE;
VAR NRO,DERNCLASSE,I,LIGNE,COLONNE,PAS,NBRECLASSE,NBCLASUJET,NBCLALIEU:INTEGER;
    ACCEPTE:SET OF '1'..'3';
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;
    REP:CHAR;
    PASCREE:BOOLEAN;
    TITRE:CHAINE;
    CLASEXIST:EXICLASSE;

 PROCEDURE RECEPCHOIX;
 BEGIN
  GOTOXY(25,6);WRITE('VOTRE CHOIX:');
  READ (REP);
  WHILE NOT(REP IN ACCEPTE)DO
    BEGIN
      GOTOXY(0,5);WRITE('ATTENTION!');
      GOTOXY(0,6);WRITE('TAPEZ SEULEMENT 1,2 OU 3!');
      GOTOXY(37,6);READ(REP)
    END
 END;(*RECEPCHOIX*)

BEGIN
  (*$R U1,U2*)
  PAGE(OUTPUT);
  ACCEPTE:=['1','2','3'];
  REP:='1';
  LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
       LINOCLASU,LINOCLALI,PASCREE);
  GOTOXY(7,0);WRITE('CREATION DE CLASSES DE MOTS');
  GOTOXY(6,1);FOR I:=1 TO 29 DO WRITE('-');
  GOTOXY(0,2);WRITE('DESIREZ-VOUS CREER:1.UNE CLASSE SUJET');
  GOTOXY(19,3);WRITE('2.UNE CLASSE LIEU');
  GOTOXY(19,4);WRITE('3:AUCUNE CLASSE');
  RECEPCHOIX;
  WHILE REP<>'3'DO
    BEGIN
```

```
      GOTOXY(0,5);
      FOR LIGNE:=5 TO 22 DO WRITELN(VIDE);
      WRITE(VIDE);
      IF NOT PASCREE
       THEN
        BEGIN
         GOTOXY(3,10);WRITE('LISTE DES CLASSES ');
         LIGNE:=12;
         COLONNE:=0;
         CASE REP OF
           '1':BEGIN
                WRITE('SUJET');
                GOTOXY(2,11);FOR I:=1 TO 25 DO WRITE('-');
                ECRCLASU(CLASEXIST,NBRECLASSE,COLONNE,LIGNE);
                NRO:=NBCLASUJET+1;
                DERNCLASSE:=NBCLASUJET;
                NBCLASUJET:=NBCLASUJET+1
               END;
           '2':BEGIN
                WRITE('LIEU');
                GOTOXY(2,11);FOR I:=1 TO 24 DO WRITE('-');
                ECRCLALI(CLASEXIST,NBRECLASSE,COLONNE,LIGNE);
                NRO:=NBCLALIEU+100+1;
                DERNCLASSE:=NBRECLASSE;
                NBCLALIEU:=NBCLALIEU+1
               END
          END(*CASE*)
         END
        ELSE
         BEGIN
          GOTOXY(0,11);
          WRITE('PAS ENCORE DE CLASSES!');
          DERNCLASSE:=0;
          CASE REP OF
           '1':BEGIN
                NBCLASUJET:=1;
                NRO:=1
               END;
           '2':BEGIN
                NBCLALIEU:=1;
                NRO:=101
               END
          END(*CASE*)
         END;
      GOTOXY(0,7);
      WRITE('LE NUMERO DE LA CLASSE EST:',NRO);
      GOTOXY(0,8);WRITE('QUEL EN EST L''INTITULE?');
      GOTOXY(0,23);WRITE('TERMINEZ VOTRE REPONSE PAR "RETURN"');
      GOTOXY(1,9);READLN(TITRE);
      NBRECLASSE:=NBRECLASSE+1;
      IF NOT PASCREE THEN FOR PAS:=NBRECLASSE DOWNTO(DERNCLASSE+2)DO
         CLASEXIST[PAS]:=CLASEXIST[PAS-1];
      CLASEXIST[DERNCLASSE+1].NUMERO:=NRO;
      CLASEXIST[DERNCLASSE+1].TITRE:=TITRE;
      PASCREE:=FALSE;
      GOTOXY(0,23);WRITE(VIDE);
      RECEPCHOIX
    END;(*WHILE REP*)
  SAVECLASSEMOT(CLASEXIST,NBRECLASSE)
END;(*CRECLASSE*)

BEGIN
END.
```

```
(*$S+*)

UNIT U12;

INTERFACE
USES (*$U *:U1.CODE*)U1,(*$U *:U2.CODE*)U2;

PROCEDURE LISTPHRASES;

IMPLEMENTATION

PROCEDURE LISTPHRASES;
VAR REP:CHAR;
    ACCEPTE:SET OF'1'..'5';
    I:INTEGER;
    CLASEXIST:EXICLASSE;
    FINLIST:BOOLEAN;

 PROCEDURE REFUSIMPRES(VAR PASCLAS,PASVERB,PASPHRA:BOOLEAN);
 BEGIN
  GOTOXY(0,10);FOR I:=1 TO 4 DO WRITELN(VIDE);GOTOXY(0,10);
  WRITE('PAS  DE PHRASES CORRECTES  ');
  GOTOXY(2,12);
  IF PASCLAS THEN WRITE('PAS DE CLASSES DE MOTS   ');
  IF PASVERB THEN WRITE('PAS DE VERBES');
  IF PASPHRA AND (NOT PASCLAS) THEN
    BEGIN
      GOTOXY(2,14);
      WRITE('PAS DE CLASSES DE MOTS')
    END
 END;(*REFUSIMPRES*)

 PROCEDURE LISPHRAECR;
 VAR I,INDS,INDV,INDL,LIGNE,NBRECLASSE,NBCLASUJET,NBCLALIEU,NBREVERBE,
        NBREMOT:INTEGER;
     SORTIE:CLASSE;
     PASCLAS,PASVERB,PASPHRA:BOOLEAN;
     REP:CHAR;
     CORRECTSEM:CORRECT;
     CLASEXIST:EXICLASSE;
     LISTNROCLASSE:SET150;
     LINOCLASU:SET1;
     LINOCLALI:SET51;
     LISTMOT:TABCAR;
     INDICE:TABIND;
     LISTVERBE:LISTE;
 BEGIN
  PAGE(OUTPUT);
  GOTOXY(7,0);WRITE('LISTE DES PHRASES CORRECTES');
  GOTOXY(6,1);FOR I:=1 TO 29 DO WRITE('-');
  LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
      LINOCLASU,LINOCLALI,PASCLAS);
  CHARGVERBE(LISTMOT,INDICE,LISTVERBE,NBREMOT,NBREVERBE,PASVERB);
  CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU,CORRECTSEM,PASPHRA);
  IF(NOT PASCLAS)AND(NOT PASVERB)AND(NOT PASPHRA)
   THEN
    BEGIN
     LIGNE:=2;
     FOR INDS:=1 TO NBCLASUJET DO
      BEGIN
       GOTOXY(0,LIGNE);
       WRITE(VIDE);
       GOTOXY(0,LIGNE);
       WRITE('SUJET:',CLASEXIST[INDS].NUMERO,':',CLASEXIST[INDS].TITRE);
       INDV:=1;
       WHILE INDV <=NBREVERBE DO
        BEGIN
         WHILE(NOT CORRECTSEM[INDS,INDV,0])AND(INDV<=NBREVERBE)DO INDV:=INDV+1;
```

```
          IF INDV<=NBREVERBE THEN
           BEGIN
            GOTOXY(0,LIGNE+1);
            WRITE(VIDE);
            GOTOXY(2,LIGNE+1);
            ECRIVERBE(LISTVERBE,LISTMOT,INDV);
            INDL:=1;
            WHILE INDL<=NBCLALIEU DO
             BEGIN
NDL+1;        WHILE(NOT CORRECTSEM[INDS,INDV,INDL])AND(INDL<=NBCLALIEU)DO INDL:=I
              IF INDL<=NBCLALIEU THEN
               BEGIN
                CHERCLASSE(CLASEXIST,NBRECLASSE,INDL+100,SORTIE);
                GOTOXY(0,LIGNE+2);
                WRITE(VIDE);
                GOTOXY(4,LIGNE+2);
                WRITE('LIEU:',SORTIE.NUMERO,':',SORTIE.TITRE);
                INDL:=INDL+1;
               END;
              IF LIGNE>20
               THEN
                BEGIN
                 LIGNE:=2;
                 CONT
                END
               ELSE LIGNE:=LIGNE+1
             END
           END;
          INDV:=INDV+1
         END;
        IF LIGNE>20
             THEN
              BEGIN
               LIGNE:=2;
               CONT
              END
             ELSE LIGNE:=LIGNE+1

       END;
      GOTOXY(0,22);WRITE(VIDE);
      GOTOXY(0,22);WRITE('FIN DE LISTE')
     END
    ELSE REFUSIMPRES(PASCLAS,PASVERB,PASPHRA);
   CONT
  END;(*LISPHRAECR*)

  PROCEDURE LISPHRAIM;
  VAR I,INDS,INDV,INDL,NBRECLASSE,NBCLASUJET,NBCLALIEU,NBREVERBE,
         NBREMOT:INTEGER;
      CLASUJET,VERBE,CLASLIEU,STRNRO:CHAINE;
      BLANC,CRRET,REP:CHAR;
      SORTIE:CLASSE;
      PASCLAS,PASVERB,PASPHRA:BOOLEAN;
      CORRECTSEM:CORRECT;
      CLASEXIST:EXICLASSE;
      LISTNROCLASSE:SET150;
      LINOCLASU:SET1;
      LINOCLALI:SET51;
      LISTMOT:TABCAR;
      INDICE:TABIND;
      LISTVERBE:LISTE;
  BEGIN
   PAGE(OUTPUT);
   GOTOXY(2,6);WRITE('IMPRESSION DE LA LISTE DES PHRASES');
   GOTOXY(15,8);WRITE('EXISTANTES');
   GOTOXY(3,11);WRITE('VERIFIEZ SI L''IMPRIMANTE EST BRANCHEE');
   GOTOXY(3,13);WRITE('PUIS TAPEZ N''IMPORTE QUELLE TOUCHE');
   READ(KEYBOARD,REP);
```

```
(*$I-*)
UNITCLEAR(6);
(*$I+*)
IF IORESULT <>0
 THEN
  BEGIN
   GOTOXY(3,15);
   WRITE('L''IMPRIMANTE N''EST PAS BRANCHEE')
  END
 ELSE
  BEGIN
   CLASUJET:='CLASSE SUJET:';
   BLANC:=' ';
   VERBE:='VERBE:';
   CLASLIEU:='CLASSE LIEU:';
   CRRET:=CHR(13);
   LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
       LINOCLASU,LINOCLALI,PASCLAS);
   CHARGVERBE(LISTMOT,INDICE,LISTVERBE,NBREMOT,NBREVERBE,PASVERB);
   CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU,CORRECTSEM,PASPHRA);
   IF(NOT PASCLAS)AND(NOT PASVERB)AND(NOT PASPHRA)
    THEN
     BEGIN
      GOTOXY(3,15);WRITE('IMPRESSION EN COURS');
      FOR INDS:=1 TO NBCLASUJET DO
       BEGIN
        UNITWRITE(6,CLASUJET[1],LENGTH(CLASUJET));
        STR(CLASEXIST[INDS].NUMERO,STRNRO);
        UNITWRITE(6,BLANC,1);
        UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
        UNITWRITE(6,BLANC,1);
        UNITWRITE(6,CLASEXIST[INDS].TITRE[1],LENGTH(CLASEXIST[INDS].TITRE));
        UNITWRITE(6,CRRET,1);
        INDV:=1;
        WHILE INDV <= NBREVERBE DO
         BEGIN
1;        WHILE(NOT CORRECTSEM[INDS,INDV,0])AND(INDV<=NBREVERBE)DO INDV:=INDV+
          IF INDV<=NBREVERBE THEN
           BEGIN
            STR(INDV,STRNRO);
            FOR I:=1 TO 2 DO UNITWRITE(6,BLANC,1);
            UNITWRITE(6,VERBE[1],LENGTH(VERBE));
            UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
            UNITWRITE(6,BLANC,1);
            I:=LISTVERBE[INDV].INDICE;
            WHILE LISTMOT[I] <>'@' DO
             BEGIN
              UNITWRITE(6,LISTMOT[I],1);
              I:=I+1
             END;
            UNITWRITE(6,CRRET,1);
            INDL:=1;
            WHILE INDL <=NBCLALIEU  DO
             BEGIN
              WHILE(NOT CORRECTSEM[INDS,INDV,INDL])AND(INDL<=NBCLALIEU)DO
                  INDL:=INDL+1;
              IF INDL<=NBCLALIEU THEN
               BEGIN
                FOR I:=1 TO 4 DO UNITWRITE(6,BLANC,1);
                UNITWRITE(6,CLASLIEU[1],LENGTH(CLASLIEU));
                CHERCLASSE(CLASEXIST,NBRECLASSE,INDL+100,SORTIE);
                STR(SORTIE.NUMERO,STRNRO);
                UNITWRITE(6,STRNRO[1],LENGTH(STRNRO));
                UNITWRITE(6,BLANC,1);
                UNITWRITE(6,SORTIE.TITRE[1],LENGTH(SORTIE.TITRE));
                UNITWRITE(6,CRRET,1);
                INDL:=INDL+1
               END
```

```
                END
              END;
            INDV:=INDV+1
          END
        END;
       GOTOXY(3,15);WRITE(VIDE);
       GOTOXY(3,15);WRITE('IMPRESSION TERMINEE')
      END
     ELSE REFUSIMPRES(PASCLAS,PASVERB,PASPHRA);
    END;(*ELSE IORESULT*)
  CONT
 END;(*LISPHRAIM*)

BEGIN
 REP:='0';
 ACCEPTE:=['1','2','3','4','5'];
 FINLIST:=FALSE;
 WHILE NOT FINLIST DO
  BEGIN
   PAGE(OUTPUT);
   GOTOXY(11,0);WRITE('MENU CONSULTATION');
   GOTOXY(10,1);FOR I:=1 TO 20 DO WRITE('-');
   GOTOXY(0,4);WRITE('1:LISTE DES PHRASES CORRECTES A L''ECRAN');
   GOTOXY(0,7);WRITE('2:LISTE DES CLASSES DE MOTS EXISTANTES');
   GOTOXY(4,8);WRITE('A L''ECRAN');
   GOTOXY(0,11);WRITE('3:IMPRESSION DE LA LISTE DES PHRASES');
   GOTOXY(4,12);WRITE('CORRECTES');
   GOTOXY(0,15);WRITE('4:IMPRESSION DE LA LISTE DES CLASSES ');
   GOTOXY(4,16);WRITE('EXISTANTES');
   GOTOXY(0,19);WRITE('5:RETOUR AU MENU GESTION DE PHRASES');
   GOTOXY(23,23);WRITE('VOTRE CHOIX:');
   READ (KEYBOARD,REP);
   WHILE NOT(REP IN ACCEPTE)DO
     BEGIN
       GOTOXY(3,21);WRITE('ATTENTION!');
       GOTOXY(3,22);WRITE('TAPEZ SEULEMENT UN NOMBRE COMPRIS');
       GOTOXY(4,23);WRITE('ENTRE 1 ET 5');
       GOTOXY(35,23);READ(REP)
     END;
   CASE REP OF '1':LISPHRAECR;
               '2':CLASSEXIST;
               '3':LISPHRAIM;
               '4':IMPRESCLAS;
               '5':FINLIST:=TRUE
            END
  END
END;(*LISTPHRASES*)

BEGIN
END.
```

```
(*$S+*)

UNIT U13;

INTERFACE
USES (*$U *:U1.CODE*)U1,(*$U *:U2.CODE*)U2;
PROCEDURE AJOUPHRASES;

IMPLEMENTATION

PROCEDURE AJOUPHRASES;
VAR COLONNE,NBRECLASSE,NBCLASUJET,NBCLALIEU,NBREVERBE,NBREMOT:INTEGER;
    PASCLAS,PASVERB,PASPHRA,FINI:BOOLEAN;
    CORRECTSEM:CORRECT;
    CLASEXIST:EXICLASSE;
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;
    LISTMOT:TABCAR;
    INDICE:TABIND;
    LISTVERBE:LISTE;

 PROCEDURE CORPAJOUPHRASES;
 VAR LIGNE,INDS,INDV,INDL:INTEGER;

  PROCEDURE AJOUSUJET(VAR INDS:INTEGER);
  VAR NROCLASSE:CHAINE;
  BEGIN
   GOTOXY(0,2);WRITE('CLASSE SUJET:');
   READLN(NROCLASSE);
   CONVERSION(NROCLASSE,INDS);
   WHILE NOT(INDS IN LISTNROCLASUJET)DO
    BEGIN
     GOTOXY(13,2);WRITE(VIDE);
     GOTOXY(3,3);WRITE('LA CLASSE ',NROCLASSE,' N''EST PAS UNE CLASSE SUJET');
     GOTOXY(13,2);READLN(NROCLASSE);
     CONVERSION(NROCLASSE,INDS)
    END;
   GOTOXY(16,2);WRITE(' ',CLASEXIST[INDS].TITRE);
   GOTOXY(30,3);WRITE('INDS:',INDS)
  END;(*AJOUSUJET*)

  PROCEDURE AJOUVERBE(VAR INDV:INTEGER);
  VAR NROVERBE:CHAINE;
  BEGIN
    GOTOXY(2,3);WRITE('VERBE:');READLN(NROVERBE);
    CONVERSION(NROVERBE,INDV);
    WHILE (INDV<=0)OR(INDV>NBREVERBE) DO
     BEGIN
      GOTOXY(3,4);WRITE('LE VERBE ',NROVERBE,' N"EXISTE PAS');
      GOTOXY(8,3);WRITE(VIDE);
      GOTOXY(8,3);READLN(NROVERBE);
      CONVERSION(NROVERBE,INDV)
     END;
    GOTOXY(2,3);
    ECRIVERBE(LISTVERBE,LISTMOT,INDV)
  END;(*AJOUVERBE*)

  PROCEDURE AJOULIEU(VAR INDL:INTEGER);
  VAR NROCLASSE:CHAINE;
      SORTIE:CLASSE;
  BEGIN
    GOTOXY(4,4);WRITE('CLASSE LIEU:');
    READLN(NROCLASSE);
    CONVERSION(NROCLASSE,INDL);
    WHILE NOT(INDL IN LISTNROCLASLIEU)DO
     BEGIN
      GOTOXY(3,5);
```

```
      WRITE('LA CLASSE ',NROCLASSE,' N''EST PAS UNE CLASSE DE LIEU');
      GOTOXY(16,4);WRITE(VIDE);
      GOTOXY(16,4);READLN(NROCLASSE);
      CONVERSION(NROCLASSE,INDL)
     END;
    CHERCLASSE(CLASEXIST,NBRECLASSE,INDL,SORTIE);
    GOTOXY(20,4);WRITE(' ',SORTIE.TITRE);
    INDL:=INDL-100
  END;(*AJOULIEU*)

 BEGIN
  COLONNE:=0;
  LIGNE:=5;
  ECRCLASU(CLASEXIST,NBRECLASSE,COLONNE,LIGNE);
  GOTOXY(0,23);WRITE('TAPEZ LE NUMERO DE LA CLASSE PUIS"RETURN"');
  AJOUSUJET(INDS);
  FOR LIGNE:=3 TO 23 DO
    BEGIN
      GOTOXY(0,LIGNE);
      WRITE(VIDE)
    END;
  COLONNE:=0;
  LIGNE:=6;
  VERBEX(LISTVERBE,LISTMOT,NBREVERBE,COLONNE,LIGNE);
  GOTOXY(0,23);WRITE('TAPEZ LE NUMERO DU VERBE PUIS"RETURN"');
  AJOUVERBE(INDV);
  CORRECTSEM[INDS,INDV,0]:=TRUE;
  FOR LIGNE:=4 TO 23 DO
    BEGIN
      GOTOXY(0,LIGNE);
      WRITE(VIDE)
    END;
  COLONNE:=0;
  LIGNE:=6;
  ECRCLALI(CLASEXIST,NBRECLASSE,COLONNE,LIGNE);
  GOTOXY(0,23);WRITE('TAPEZ LE NUMERO DE LA CLASSE PUIS"RETURN"');
  AJOULIEU(INDL);
  CORRECTSEM[INDS,INDV,INDL]:=TRUE;
  FOR LIGNE:=6 TO 23 DO
   BEGIN
    GOTOXY(0,LIGNE);
    WRITE(VIDE)
   END;
  CONTARRET(2,15,FINI)
 END;(*CORPAJOUPHRASES*)

BEGIN
 PAGE(OUTPUT);
 GOTOXY(7,0);WRITE('AJOUT DE PHRASES CORRECTES');
 GOTOXY(6,1);FOR COLONNE:=1 TO 28 DO WRITE('-');
 LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
      LINOCLASU,LINOCLALI,PASCLAS);
 CHARGVERBE(LISTMOT,INDICE,LISTVERBE,NBREMOT,NBREVERBE,PASVERB);
 CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU,CORRECTSEM,PASPHRA);
 FINI:=FALSE;
 IF(NOT PASCLAS)AND(NOT PASVERB)
  THEN
   BEGIN
    WHILE NOT FINI DO CORPAJOUPHRASES;
    SAUVEPHRASECORRECTES(CORRECTSEM,NBCLASUJET,NBREVERBE,NBCLALIEU)
   END
  ELSE
   BEGIN
    GOTOXY(2,12);
    IF PASCLAS THEN WRITE('PAS DE CLASSES DE MOTS! ');
    IF PASVERB THEN WRITE('PAS DE VERBES!');
    CONT
   END
```

```
END; (*AJOUPHRASES*)

BEGIN
END.
```

```
(*$S+*)

UNIT U14;

INTERFACE
USES (*$U *:U1.CODE*)U1,(*$U *:U2.CODE*)U2;
PROCEDURE SUPPRESPHRASES;

IMPLEMENTATION

PROCEDURE SUPPRESPHRASES;
VAR I,COLONNE,NBRECLASSE,NBCLASUJET,NBCLALIEU,NBREVERBE,NBREMOT:INTEGER;
    PASCLAS,PASVERB,PASPHRA,FINI:BOOLEAN;
    CORRECTSEM:CORRECT;
    CLASEXIST:EXICLASSE;
    LISTNROCLASSE:SET150;
    LINOCLASU:SET1;
    LINOCLALI:SET51;
    LISTMOT:TABCAR;
    INDICE:TABIND;
    LISTVERBE:LISTE;

 PROCEDURE CORPSUPPRESPHRASES;
 VAR LIGNE,INDS,INDV,INDL:INTEGER;

  PROCEDURE SUPVERBE(VAR INDV:INTEGER);
  VARNROVERBE:CHAINE;
  BEGIN
    GOTOXY(2,3);WRITE('VERBE:');
    READLN(NROVERBE);
    CONVERSION(NROVERBE,INDV);
    WHILE (INDV=0)OR(INDV>NBREVERBE) DO
     BEGIN
      GOTOXY(3,4);WRITE('LE VERBE ',NROVERBE,' N''EXISTE PAS');
      GOTOXY(8,3);WRITE(VIDE);
      GOTOXY(8,3);READLN(NROVERBE);
      CONVERSION(NROVERBE,INDV)
     END;
    GOTOXY(2,3);
    ECRIVERBE(LISTVERBE,LISTMOT,INDV)
  END;(*SUPVERBE*)

  PROCEDURE SUPSUJET(VAR INDS:INTEGER);
  VAR NROCLASSE:CHAINE;
  BEGIN
    GOTOXY(0,2);WRITE('CLASSE SUJET:');
    READLN(NROCLASSE);
    CONVERSION(NROCLASSE,INDS);
    WHILE NOT(INDS IN LISTNROCLASUJET)DO
     BEGIN
      GOTOXY(13,2);WRITE(VIDE);
      GOTOXY(3,3);WRITE('LA CLASSE ',NROCLASSE,' N''EST PAS UNE CLASSE SUJET');
      GOTOXY(13,2);READLN(NROCLASSE);
      CONVERSION(NROCLASSE,INDS)
     END;
    GOTOXY(16,2);WRITE(' ',CLASEXIST[INDS].TITRE);
    GOTOXY(35,2);WRITE('INDS:',INDS)
  END;(*SUPSUJET*)

  PROCEDURE SUPLIEU(VAR INDL:INTEGER);
  VAR NROCLASSE:CHAINE;
      SORTIE:CLASSE;
  BEGIN
    GOTOXY(4,4);WRITE('CLASSE LIEU:');
    READLN(NROCLASSE);
    CONVERSION(NROCLASSE,INDL);
    WHILE NOT(INDL IN LISTNROCLASLIEU)DO
     BEGIN
```

```
      GOTOXY(3,5);
      WRITE('LA CLASSE ',NROCLASSE,' N''EST PAS UNE CLASSE DE LIEU');
      GOTOXY(16,4);WRITE(VIDE);
      GOTOXY(16,4);READLN(NROCLASSE);
      CONVERSION(NROCLASSE,INDL)
     END;
    CHERCLASSE(CLASEXIST,NBRECLASSE,INDL,SORTIE);
    GOTOXY(20,4);WRITE(' ',SORTIE.TITRE);
    INDL:=INDL-100
  END;(*SUPLIEU*)

 BEGIN
  COLONNE:=0;
  LIGNE:=5;
  ECRCLASU(CLASEXIST,NBRECLASSE,COLONNE,LIGNE);
  GOTOXY(0,23);WRITE('TAPEZ LE NUMERO DE LA CLASSE PUIS"RETURN"');
  SUPSUJET(INDS);
  FOR I:=3 TO 23 DO
    BEGIN
     GOTOXY(0,I);
     WRITE(VIDE)
    END;
  COLONNE:=0;
  LIGNE:=6;
  VERBEX(LISTVERBE,LISTMOT,NBREVERBE,COLONNE,LIGNE);
  GOTOXY(0,23);WRITE('TAPEZ LE NUMERO DU VERBE PUIS"RETURN"');
  SUPVERBE(INDV);
  FOR I:=4 TO 23 DO
   BEGIN
    GOTOXY(0,I);
    WRITE(VIDE)
   END;
  COLONNE:=0;
  LIGNE:=6;
  ECRCLALI(CLASEXIST,NBRECLASSE,COLONNE,LIGNE);
  GOTOXY(0,23);WRITE('TAPEZ LE NUMERO DE LA CLASSE PUIS"RETURN"');
  SUPLIEU(INDL);
  CORRECTSEM[INDS,INDV,0]:=FALSE;
  CORRECTSEM[INDS,INDV,INDL]:=FALSE;
  FOR INDL:=1 TO NBCLALIEU DO
   IF CORRECTSEM[INDS,INDV,INDL] THEN CORRECTSEM[INDS,INDL,0]:=TRUE;
  CONTARRET(2,23,FINI)
 END;(*CORPSUPPRESPHRASES*)

BEGIN
 PAGE(OUTPUT);
 GOTOXY(7,0);WRITE('SUPPRESSION DE PHRASES CORRECTES');
 GOTOXY(6,1);FOR I:=1 TO 33 DO WRITE('-');
 LOADCLASSEMOT(CLASEXIST,NBRECLASSE,NBCLASUJET,NBCLALIEU,LISTNROCLASSE,
    LINOCLASU,LINOCLALI,PASCLAS);
 CHARGVERBE(LISTMOT,INDICE,LISTVERBE,NBREMOT,NBREVERBE,PASVERB);
 CORRECTIONPHRASES(NBCLASUJET,NBREVERBE,NBCLALIEU,CORRECTSEM,PASPHRA);
 IF (NOT PASCLAS) AND (NOT PASVERB)
  THEN
   BEGIN
    FINI:=FALSE;
    WHILE NOT FINI DO CORPSUPPRESPHRASES;
    SAUVEPHRASECORRECTES(CORRECTSEM,NBCLASUJET,NBREVERBE,NBCLALIEU)
   END
  ELSE
   BEGIN
    GOTOXY(2,12);
    IF PASCLAS THEN WRITE('PAS DE CLASSES DE MOTS! ');
    IF PASVERB THEN WRITE('PAS DE VERBES!');
    CONT
   END
END;(*SUPPRESPHRASES*)
```

```
BEGIN
END.
```

A N N E X E 17.

EXEMPLES DE PHRASES GENEREES PAR LE MODULE DE GENERATION DE PHRASES.

- EL HUESPED NADAR EN NUESTRAS MARES.
  L'hôte nage dans nos mers.
- ESE RATON ESTAR EN MERCADOS.
  Cette souris est dans des marchés.
- EL JARRO ESTAR EN ESE GARAGE.
  Le pot est dans ce garage.
- ESTE PERRO ESTAR EN TEATROS.
  Ce chien est dans des théâtres.
- CARNES SITUARSE EN SUS COMODAS.
  Des viandes se trouvent dans vos/leurs commodes.
- AQUEL CARBON SITUARSE EN TUS COMODAS.
  Ce charbon se trouve dans tes commodes.
- LOS BURROS COMER EN EL TASCA.
  Les ânes mangent dans le bistrot.
- ESTOS RATONES ESTAR EN ESOS CINES.
  Ces souris sont dans ces cinémas.
- ESTE COMISARIO VENIR A AQUELLA CIUDAD.
  Ce commissaire vient à/dans cette ville.
- LOS EMPLEADOS MARCHARSE A NUESTROS EDIFICIOS.
  Les employés s'en vont / viennent vers nos immeubles.
- LA TORTILLA SITUARSE EN COMODAS.
  La tortilla se trouve dans/sur des commodes.
- MIS BURROS QUEDAR EN NUESTRAS CARCELES.
  Mes ânes restent en nos prisons.
- EL BURRO SITUARSE EN LOS MONTES.
  L'âne se trouve dans les monts.
- ESTAS CARNES SITUARSE EN TALLERES.
  Ces viandes se trouvent dans des ateliers.
- MIS FAMILIAS MORIR EN ESTA PROVINCIA.
  Mes familles meurent dans cette province.
- ESE COMISARIO VENIR A ESE PLAYA.
  Ce commissaire vient à cette plage.
- EMPLEADOS MORIR EN ESE BARRIO.
  Des employés meurent dans ce quartier.

- ESTAS AMIGOS QUEDAR EN SUS PISOS.
  Ces amis restent dans leurs/vos appartements.
- EL DOCTOR TRABAJAR EN CINES.
  Le docteur travaille dans des cinémas.
- ESA TORTILLA SITUARSE EN ESTA COMODA.
  La tortilla est dans cette commode.
-ESTOS DUENOS VIVIR EN EDIFICIOS.
  Ces maîtres vivent dans des immeubles.
- ESTE AUTOBUS VENIR A AQUELLA AVENIDA.
  Cet autobus vient/va à cette avenue.
- EL PERRO ESTAR EN AQUELLA PROVINCIA.
  Le chien est dans cette province.
- NUESTROS JARROS ESTAR EN ESOS APARADORES.
  Nos pots sont dans ces commodes.
- EL HUESPED IR A VUESTRAS HABITACIONES.
  L'hôte va à/dans vos chambres.
- ASESINOS VENIR A LAS AVENIDAS.
  Des assassins viennent aux avenues.
- MANTILLAS QUEDAR EN VUESTROS CINES.
  Des mantilles restent dans vos cinémas.
- LA ARTISTA SITUARSE EN ESTAS SILLAS.
  L'artiste se trouve sur ces chaises.
- VUESTROS GATOS SITUARSE EN SUS COCINAS.
  Vos chats sont dans leurs/vos cuisines.
- ESTE RATON ESTAR EN AQUEL PUENTE.
  Cette souris est sur ce pont.
- ESE BURRO NADAR EN LA COSTA.
  Cet âne nage sur la côte.
- LOS AMIGOS QUEDAR EN SUS MERCADOS.
  Les amis restent dans leurs/vos marchés.
- AQUEL CHURRO SITUARSE EN ESTE TALLER.
  Ce beignet est dans cet atelier.
- LAS CARNES SITUARSE EN MIS TALLERES.
  Les viandes sont dans mes ateliers.
- AUTOBUS QUEDAR EN LA CIUDAD.
  Des autobus restent dans la ville.
- SOMBREROS QUEDAR EN CARCELES.
  Des chapeaux restent dans les prisons.
- ESTE CHURRO SITUARSE EN LA COMODA.
  Ce beignet se trouve dans/sur la commode.

- BURROS SUBIR A AQUELLOS PLAYAS.
  Des ânes montent à ces plages.
- LA AMIGA QUEDAR EN LAS TASCAS.
  L'amie reste dans les bistrots.
- ASESINOS SITUARSE EN AGUAS.
  Des assassins se trouvent dans les eaux.
- LOS DUENOS TRABAJAR EN BARRIOS.
  Les maitres travaillent dans des quartiers.
- AQUELLOS HUESPEDES VENIR A LOS DESVANES.
  Ces hôtes viennent dans les greniers.
- ESE ARTISTA SITUARSE EN BRUSELAS.
  Cette artiste se trouve dans Bruxelles.
- TUS TORTILLAS SITUARSE EN LA COMODA.
  Tes tortillas sont dans la commode.
- LA TORTILLA SITUARSE EN TALLERES.
  La tortilla se trouve dans des ateliers.
- LOS COMISARIOS MARCHARSE A EL MOLINO.
  Les commissaires s'en vont au moulin.
- UN CARBON SITUARSE EN EL TALLER.
  Un charbon se trouve dans l'atelier.
- ESTE EMPLEADO MORIR EN ESE MERCADO.
  Cet employé meurt dans ce marché.
- ESTOS AUTOBUSES ESTAR EN SUS PUEBLOS.
  Ces autobus sont dans leurs/vos villages.
- ESE EMPLEADO METERSE EN PUERTOS.
  Cet employé se met/fourre dans des ports.
- AMIGOS QUEDAR EN LA CARCEL.
  Des amis restent en prison.
- TUS SOMBREROS QUEDAR EN EL GARAGE.
  Tes chapeaux restent dans le garage.
- NUESTRAS AGUAS SITUARSE EN BRUSELAS.
  Nos eaux sont dans Bruxelles.
- EL COMISARIO QUEDAR EN LAS CASAS.
  Le commissaire reste dans les maisons.